Urban mobility : leveraging machine learning and data masses for the building of simulators

Gauthier Lyan

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Gauthier LYAN

Mobilité urbaine : Apprentissage automatique pour la construction de simulateurs à l'aide de masses de données.

Urban mobility : Leveraging machine learning and data masses for the building of simulators

Thèse présentée et soutenue à L'IRISA, le 23 septembre 2021
Unité de recherche : Equipes DIVERSE/DRUID, IRISA

Rapporteurs avant soutenance :

Neila Bhouri — Chercheuse HDR à l'université Gustave Eiffel, Marne la vallée, FRANCE
Karine Zeitouni — Professeure à l'université de Versailles Saint-Quentin-en-Yvelines, Versailles, FRANCE

Composition du Jury :
Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	François BODIN	Professeur à l'université de Rennes 1, Rennes, FRANCE
Examinateurs :	Genoveva VARGAS-SOLAR	Directrice de recherche au CNRS, LIRIS, Villeurbanne, FRANCE
	Neila Bhouri	Chercheuse HDR à l'université Gustave Eiffel, Marne la vallée, FRANCE
	Karine Zeitouni	Professeure à l'UVSQ, Versailles, FRANCE
Dir. de thèse :	Jean-Marc JEZEQUEL	Professeur, Université Rennes 1, Rennes, France
Co-dir. de thèse :	David GROSS-AMBLARD	Professeur, Université Rennes 1, Rennes, France

Invité(s) :

Anne STRUGEON — Directrice qualité, Keolis Rennes, Rennes, FRANCE

Remerciements

Je me suis longtemps cru incapable de réussir dans les études supérieures. Toute ma primaire et mon secondaire, j'ai été l'élève qui peut mieux faire, qui ne va jamais au bout des choses, celui qui préfère vivre que s'échiner au travail pour un bonus de points potentiellement négligeable. Il n'a jamais s'agit que d'une banale question de coefficient bénéfice/investissement. Pendant des années, cette idée est restée ancrée en moi comme une bernique sur son rocher. Jusqu'à un certain soir, chez Florian CADET qui, lors d'une partie de Beat Hazard™ particulèrement ardue, me fit remarquer que j'avais tendance à fuir toute forme de challenge, me pensant incapable d'arriver à sortir de ma zone de confort. Merci, Flo, pour ce déclic (et pour tout le reste), auquel je repense à chaque fois qu'un défi me fait face. Merci aux quelques enseignants qui ont su m'éviter une dérive scolaire, je pense à Mr AYMONIN, Mme MUGNIER, Mme DUNAN, Mr ROUSSEAU, ... Sans eux, nul doute que ces lignes n'auraient jamais été écrites. Merci aux autres d'avoir fait germer en moi l'envie de leur donner tort: "Malgré" un bac ES et 3 de moyenne en maths en terminale, j'ai su me refaire, seul, pour arriver jusuqu'ici. OVNI parmi les OVNIs, je dois une partie de ce parcours atypique à la grande famille de la MIAGE, que je suis fier de représenter. OUI, un miagiste peut faire de la recherche et devenir docteur en sciences. Aux miagistes qui liraient ces lignes, allez-y, vous avez ce qu'il faut pour y arriver, vous avez peut-être même deux ou trois trucs en plus, vous voyez où je veux en venir.

Papa, maman, merci d'avoir cru en moi jusqu'au bout, j'ai bien conscience d'avoir une chance que beaucoup d'autres non pas.

Merci à Thomas GOEURY, éternel parrain, qui m'a montré la voie du doctorat, ta sagesse est d'or.

Merci à Jacques BLOCKLET, tu as été mon encyclopédie de l'ingénierie des transports, sans toi je n'aurais probablement pas su résoudre nombre de problèmes. Il ne fait aucun doute que l'existence de cette thèse est en grande partie de ton fait.

Alison, merci d'avoir partagé ton bureau avec moi jusqu'à cette maudite pandémie. On se sera bien marrés, et je te souhaite le meilleur pour la suite (bonjour de la part de

Jean-Michel). Béa, merci pour les cafés et les pauses potager, ça pousse, ça pousse!

Merci à vous deux, David et Jean-Marc, pour les conseils, la confiance et la liberté de travail qui ont, de concert, abouti au succès de cette thèse. Vous avez été d'une grande aide et nul doute que vos enseignements me seront utiles au quotidien dans les temps futurs.

Merci à Anne et Sébastien pour leur suivi et leurs encouragements. Malgré toutes les difficultés rencontrées durant notre parcours, votre accompagnement et vos encouragements ont été d'importants soutiens durant cet exercice.

Merci à Keolis Rennes de m'avoir accueilli et d'avoir participé au bon déroulement de cette thèse riche en expériences. Merci aussi pour l'intérêt porté aux travaux et à la volonté d'inscrire ces derniers dans la durée.

Un grand merci à Neila BHOURI et Karine ZEITOUNI pour le travail de relecture sur ce manuscrit, permettant la qualification de mes travaux. Votre expertise et vos regards sur cette thèse sont précieux.

Merci aux vulgarisateurs scientifiques qui font un boulot fabuleux pour transmettre des connaissances sourcées, fiables et utiles au plus grand nombre, malgré le temps de travail phénoménal (cf. loi de BRANDOLINI, effet DUNNING-KRUGER). La transmission de la connaissance est nécessaire, mais l'art requis pour le faire est très loin d'être accessible à tous, je l'ai appris à mes dépens.

Merci à ces trois années, les plus riches de ma vie intellectuellement parlant. Trois années, qui, arrivées à leur terme, me poussent à dire que PASCAL et SOCRATE avaient raison: Aujourd'hui, le silence éternel de ces espaces infinis m'effraie tout autant que l'immensité insondable de mon ignorance. Puissent ces frayeurs demeurer, bien que la température aille inévitablement croissante.

Table of Contents

INTRODUCTION EN FRANÇAIS

Au cours des dernières décennies, la masse de données mondiale n'a eu de cesse de prendre de l'ampleur [30] [1]. Dans nombre de domaines pour lesquels l'Informatique est un support d'aide à la décision, comme le transport, l'industrie ou même la recherche, les données sont devenues un outil crucial et stratégique. Avec une croissance continue des données générées annuellement ainsi que des capacités de stockage, les nouvelles données désormais massives avec l'émergence du Big Data sont devenues une mine d'or pour quiconque dispose des capacités et du matériel nécessaires pour en extraire des informations. En particulier, l'existence de données diverses et interopérables, notamment sur les villes intelligentes, rend raisonnable l'utilisation d'outils d'apprentissage automatique pour l'aide à la décision de problèmes jusqu'alors difficiles à modéliser.

Cependant, il faut savoir qu'il existe des verrous techniques et scientifiques qui rendent délicates ce type d'approche. Tout d'abord, les données provenant du monde réel (c'est à dire des données non-synthétiques) sont rarement utilisables sans pré-traitements [49]. Les valeurs manquantes, erronées, dupliquées ou même aberrantes au coeur des jeux de données sont autant d'exemples communs à de nombreuses sources de données issues du monde réel, que ce soit l'industrie, les services publics, etc. De plus, la fusion de deux ou plusieurs sources de données souvent nécessaire pour consolider les jeux de données n'est en général pas exempte de complexité technique car les données de ces différentes sources peuvent diverger dans leur format, leur encodage, leur langage, etc. L'ensemble des verrous techniques évoqués ci-avant justifient à eux seuls le coût potentiellement élevé de ces traitements, que ce soit en termes de temps de calcul (pour du prétraitement ou de la conception), mais aussi en temps de travail humain, et donc, pécunier. [41, 57, 60, 70]. Deuxièmement, des problèmes de capacités de stockage peuvent se poser pour les projets qui nécessitent l'usage de masses de données conséquentes. Il n'est en effet plus rare d'avoir à traiter des gigaoctets de données pour une simple tâche d'apprentissage automatique [21] qui, une fois enrichis, nécessitent en retour au moins autant d'espace de stockage pour être correctement sauvegardés et utilisés pour répondre à diverses problématiques métier. Troisièmement, et pour finir, la manne de personnes disposant des connaissances et savoir-

1. Statista Digital Economy Compass 2019

faire techniques nécessaires pour rassembler, traiter et extraire les informations de ces masses de données est faible [46]. En effet, les *data scientists* et autres spécialistes de la donnée sont des ingénieurs / chercheurs de haut niveau dont la formation exige de 5 à 8 années à temps plein, ce qui restreint d'autant le nombre potentiel de candidats à l'embauche.

Dans les domaines industriels pour lesquels l'Informatique possède en premier lieu une fonction de support, il existe un problème que nous appellerons "le problème de l'avalanche de données". Ce problème se caractérise par l'existence de systèmes d'information de plus en plus complexes qui génèrent des quantités de données qui croissent à minima linéairement dans le temps. Ajoutons à cela que ces systèmes d'information sont souvent maintenus en condition opérationnelle par une petite équipe d'ingénieurs et techniciens Informatiques spécialisés qui y consacrent tout leur temps de travail, les rendant indisponible pour des tâches supplémentaire d'exploitation des données. Cette situation tend à rendre l'utilisation des données que l'entreprise possède tout à fait impossible sans l'intervention de personnels experts de la données ayant accès à du matériel performant et un espace de stockage correctement proportionné au regard de la masse de données générée par l'entreprise.

Les transports publics sont typiquement un domaine industriel faisant un usage important d'outils informatisés en tant que support de leurs activités. Les réseaux de transports publics sont de plus en plus connectés et intégrés dans les villes et métropoles qui les accueillent. En effet, les métros, les tramways, les bus, etc. sont surveillés 24 heures sur 24, 7 jours sur 7, grâce à de nombreux capteurs embarqués ou sur le terrain et qui produisent des données en grande quantité et de façon quasi continue [41]. Ces données sont variées et composées de multiples indicateurs tels que la vitesse des véhicules, le temps de passage à un arrêt/station, les informations d'embarquement/débarquement ou même la consommation de carburant. Il est de plus en plus commun que les villes soient amenées à utiliser tout ou partie de ces données afin de fournir des applications web ou smartphone aux voyageurs proposant des outils de planification des déplacements, en leur donnant des informations telles que, par exemple, les horaires théoriques et l'heure d'arrivée prévue du prochain bus/metro/tram à la station où ils vont attendre [36]. Cependant, les systèmes d'information des transports publics sont souvent complexes et composés de multiples logiciels fonctionnant en silo, et donc indépendamment les uns des autres. Ceci rend leur maintenance techniquement complexe et exigeante en termes de ressources humaines et d'interfaçage. De plus, si les données qu'ils génèrent semblent être théoriquement in-

teropérables , le coût et la charge de travail nécessaire à l'enrichissement de ces données afin d'obtenir un jeu de données consolidées peut-être rapidement étouffants [41, 57].

Prenons l'exemple d'un réseau de bus urbain, celui-ci fournit des données à différents niveaux de granularité dans les dimensions spatiales (ex. ligne de bus) et temporelles (ex. heure de pointe du soir), allant par exemple du passage d'un bus à un arrêt en temps réel à la vitesse moyenne d'une ligne de bus sur une année. Au vu de la quantité et de la richesse des données générées, on pourrait être tenté de les exploiter pour accélerer la prise de décisions ou effectuer des analyses stratégiques dans l'objectf d'améliorer les performances du réseau de bus. Toutefois cela exige de se confronter au problème de l'avalanche de données, et donc de trouver un moyen de traiter efficacement ces données pour les intégrer dans un outil d'aide à la décision, qui devra lui-même être capable de fournir des moyens d'exploiter ces données.

Si le nettoyage des données est un problème qui n'est pas généralisable parce que spécifique à chaque source de données et cas d'utilisation, on peut s'attendre à ce qu'il existe des moyens de faciliter l'intégration des données spatio-temporelles pour l'analyse stratégique et la prise de décision. Par exemple, en créant un cadre technique qui permette l'intégration transparente d'outils d'analyse statistique des données passées et temps réel, mais aussi de modèles prédictifs pour aider la planification et la prise de décisions stratégiques.

Contributions

Cette thèse propose quatre contributions principales:

Tout d'abord, nous affirmons que le nettoyage des données est nécessaire si l'on veut s'assurer que les décisions prises au travers d'outils support le soient via des données véritablement représentatives du monde réel. De fait, selon les besoins de l'utilisateur, différents niveaux de nettoyage des données peuvent être utilisés, en fonction d'un seuil établi sur la base d'un compromis coût de calcul/niveau de qualité de la donnée souhaité. Pour ce faire, nous avons analysé et qualifié les données générées par le réseau de bus STAR de Rennes Metropole puis nous avons évalué l'impact des variations de la qualité des données sur la précision des modèles prédictifs. Les ensembles de données massifs issus du SI du réseau de bus STAR, que nous avons enrichis par l'usage de règles de gestion métier et par croisement avec des jeux de données internes complémentaires ont été utilisés pour mener ces expérimentations. Les résultats obtenus montrent que l'augmentation de la

qualité des données et la réduction du bruit améliorent la précision des modèles prédictifs, mais qu'une sur-qualification des jeux de données est contre productive car chère en coût de calcul, et n'améliorant pas ou peu la précision des modèles prédictifs.

Deuxièmement, nous analysons un ensemble de facteurs qui sont connus pour avoir un impact sur la vitesse commerciale des bus tels que le trafic routier, l'infrastructure routière, la fréquentation, etc. Nous évaluons l'existence de ces facteurs à l'aide de données hétérogènes massives, et levons une série d'hypothèses sur la pertinence de ces facteurs. En enrichissant les données issues du réseau de bus STAR avec des jeux de données exogènes incluant l'infrastructure routière, le matériel des bus, la météo et la fréquentation, nous avons mis en lumière les interactions entre la vitesse commerciale des bus et l'environnement, et détecté les caractéristiques importantes pour l'apprentissage automatique. Nous avons pu suggérer l'impact de certains de ces facteurs, notamment grâce aux données issues du confinement de 2020, ayant généré une situation unique sur le réseau avec des variables proches de 0 (traffic et fréquentation). Ceci aboutissant à la levée d'un certain nombre d'hypothèses permettant aux opérateurs de réseau de bus de mieux maîtriser la vitesse commerciale des bus, et donc de maintenir le réseau de bus attractif.

Troisièmement, nous proposons une approche générique d'apprentissage automatique multi-niveau pour prédire la vitesse sur tout ou partie de nouvelles lignes de bus, pour lesquelles aucune donnée n'existe. Cette approche exploite les caractéristiques spatio-temporelles des ensembles de données du réseau de bus, que l'on modélise sous forme de multi-graphes temporels et fait appel aux règles d'agrégation inhérentes à ce modèle. Ceci permet de prédire la vitesse des bus au niveau des inter-arrêts (qui consistent en deux arrêts de bus contigus sur une ligne de bus) jusqu'aux lignes de bus entières en agrégeant les prédictions des inter-arrêts. Ici, notre objectif est de proposer une méthode permettant d'estimer de manière fiable et réaliste quelle vitesse commerciale il est possible d'atteindre sur n'importe quel endroit du réseau de bus et à n'importe quelle échelle, y compris pour de nouvelles lignes de bus. Ces travaux font appel aux résultats des deux contributions précédentes et ont permis de mettre en évidence que 1) les modèles micro-prédictifs appliqués à la prédiction de vitesse des bus proposent non seulement une méthode permettant de prédire la vitesse sur le réseau sans contrainte géographique ou temporelle (possibilité de prédire la vitesse sur tout ou partie de lignes de bus), mais qu'en plus 2) il est possible de les utiliser pour prédire la vitesse de lignes de bus pour lesquelles aucune donnée n'existe, avec une précision égalant au moins les méthodes traditionnelles

à gros grain (i.e à l'échelle d'une ligne de bus complète).

Notre dernière contribution est le résultat des trois premières et consiste en un cadre technique qui permet l'intégration des données massives conjointement avec des outils d'analyse et de prédiction. L'objectif alors défendu est de rendre l'apprentissage automatique et l'analyse statistique facilement accessibles aux utilisateurs non spécialisés, tout en suggérant que les systèmes multi-outils non intégrés utilisés pour la gestion des données vont devenir caducs. Nous proposons un cadre technique de manipulation des données appelé DataTime, basé sur les trois contributions précédentes de cette thèse, qui intègre le passé, le présent et le futur dans un système d'information traditionnel contenant des modèles a priori. A travers une application de ce concept via des développements en Scala, et l'exploitation des données acquises jusqu'alors, un outil a été conçu pour démontrer que l'intégration des données et des dimensions spatio-temporelles dans un seul utilitaire est possible. Cet outil permet la manipulation des données dans l'espace (réseau de bus, à n'importe quelle granularité), et dans le temps (passé, présent, futur) en intégrant directement et de manière transparente les modèles de gestion des données, d'analyse, et de prédiction. Cette dernière contribution contient la revendication principale que cette thèse défend, à savoir que l'intégration de divers ensembles de données hautement qualifiées provenant du monde réel dans un modèle spatio-temporel unique offre un moyen qualitatif, efficace et peu coûteux de faire des analyses, des prédictions et d'aider à la prise de décisions stratégiques pour les réseau de bus.

Liste des publications en date du 23 septembre 2021

— "**Impact of data cleansing on bus commercial speed prediction**", accepté dans le journal Springer-Nature computer science, en révision mineure.
— "**On the Quality of Compositional Prediction for "What If" Analytics on Graphs**" Accepté chez MLKGraphs Workshop 2021, présenté en septembre 2021.
— "**DataTime: a Framework to smoothly Integrate Past, Present and Future into Models**" Accepté chez MODELS21, présenté en octobre 2021.

INTRODUCTION

Over the last decades, there has been more and more data generated all over the world [30] [2]. In every domain for which computer science could help decision making, such as transportation, industry or even research, data has become a crucial and strategical tool. With a continuous growth of yearly generated data while the storage capacity of hardware was growing too, the newly massively available data of the Big Data era has become a gold mine for anyone who has the capabilities and the hardware to extract information out of it. In particular, the existence of various and inter-operable data including smart-cities [50] makes the use of machine learning focused tools reasonable for decision helping systems.

However, one must know that there are obstacles that make the creation of integrated decision tools quite tricky to achieve. Firstly, real-world data is dirty [49]. Missing, wrong, duplicated or even aberrant values are as many examples of dirty information that can be found in real world data. Also, merging two or more data sources is a harsh task because they can diverge in their format, encoding, language, etc. Thus the merging becomes costly in terms of computing time or pre-processing/conception [41, 57, 60, 70]. Secondly, there might be a storage issue for project that need large scale datasets. One can quickly have to process gigabytes of data for a simple machine learning task or even terabytes of data [21] that require at least as much storage to be kept for other purposes such as business or research. Thirdly, there are few people who have the knowledge that is necessary to gather, process and extract information of this massively available data [46]. Indeed, experts like data scientists are high-level engineers / researchers for which education takes 5 to 7 years to complete, yielding a very tense employment market in this specific domain.

In industrial fields that are not computer-science focused there is an issue that we call "the data overwhelming problem" which consist of having more and more complex information systems that generate more and more data with a small team of IT engineers who are already busy keeping the information system in good condition. Thus the use of data that the company owns becomes quite often a lost opportunity.

One example of such an industrial field is the public transportation domain. Public

2. Statista Digital Economy Compass 2019

transport are more and more interlaced with smart cities. Indeed subways, tramways, bus, etc are monitored 24/7 through numerous sensors that yield data [41]. This data is quite various, including speed, dwell time at a stop/station, boarding / unboarding information or even fuel consumption. Moreover, cities often use this data to furnish web or smartphone applications that are used by travelers to plan their trips, giving them information such as time of arrival of the next shuttle at the station they will wait at [36]. However, those information systems are often complex and made of multiple software that are independent from one another, making their maintenance tricky and demanding in terms of human resources. Thus, the data they generate usually suffer from the problem of being theoretically inter-operable, yet in practice this can become costly to achieve [41, 57].

Lets take a bus network as an example. Such a network yields data at different level of granularity in both space and time dimensions, e.g from a bus passing at a bus stop in real time to the average speed of a bus line over a year. Given the amount and richness of the data generated, it might be tempting to use it for decision making or strategic analysis with the objective of improving the performance of the bus network. However, this requires to face the data overwhelming problem, and thus to find a way to efficiently process this data to integrate it in a decision support tool.

If data cleansing is a problem that can not be fully generalized because it is specific to each data source and use case[49], we can expect that there are means to facilitate the use of spatio-temporal data for strategical analysis for such a case by creating a framework that seamlessly embeds analysis tools like statistical analysis of past data and real time observations; but also predictions for strategical decisions and planning using predictive models.

Contributions

First, we claim in chapter 2 that data cleansing is needed if one wants to make sure that decisions are made over data that represent what actually happens in the real world. Depending on the need of the user, different levels of data cleansing can be leveraged, based on a threshold built over a computing cost / desired quality trade-off.

Second, in chapter 3, we analyze a set of factors such as traffic, road infrastructure, ridership, etc. that are told to be impacting the bus commercial speed. We assess the existence of those factors with the help of massive heterogeneous data, and finally raise a

series of hypotheses about those factors.

Third, we propose a generic multilevel machine learning approach to predict the speed on all or part of new bus lines, for which no data exists, in chapter 4. This approach exploits the spatio-temporal characteristics of the bus network datasets, which are hereby modeled as a temporal multi-graph; and makes use of the aggregation rules inherent in this model. This allows the predicting of bus speeds at inter-stations (which consist of two contiguous bus stops on a bus line) up to entire bus lines by aggregating the predictions of the inter-stations. Here, our goal is to propose a method to reliably and realistically estimate what commercial speed can be achieved at any location in the bus network and at any scale, including new bus routes.

Chapter 5 describes our last contribution, which is the result of the first three. It consists of a technical framework that allows the integration of massive data with analysis and prediction tools. The goal is to make machine learning and statistical analysis easily accessible to non-specialized users, while suggesting that non-integrated multi-tool systems used for data management will become obsolete.

Finally we defend the thesis that the integration of various highly qualified real world dataset in a single spatio-temporal model offers an efficient, cheap and qualitative way to seamlessly make analysis, predictions and strategical decisions for bus networks, deprecating the use of non-integrated multi-tools systems for data management.

Chapter 1

State of the art

Establishing the state of the art of the scientific domains to which this manuscript belongs is a mandatory first step towards the presentation of our contributions. Its aim is to highlight what problems exist, and why our contributions propose approaches to tackle them.

1.1 Context and motivation

1.1.1 Context

Keolis Rennes is the company that manages the Urban Public Transportation Network (UPTN) of the city of Rennes, France. This network, named STAR [1], is based on a central subway line, and a wide bus network that serves both the city of Rennes and all the suburban areas. In total, the bus network covers more than $550km^2$ The company manages a total of 116 bus lines over which more than 600 buses can be traveling during rush hours. The bus network information system is made of several sub systems including an Automatic Vehicle Location (AVL) that yields large amount of fine-grain data (inter-station) both in real and delayed time.

Industrial fields that are not computer-science focused, like public transportation, often face an issue that we could call "the data overwhelming problem". This consists of having more and more complex information systems that generate more and more data, with a small team of IT engineers who are already too busy keeping the information system healthy. Thus the management and the use of the massive data that the company owns becomes quite impossible. In companies like Keolis Rennes that rely on stable and well known technologies such as standard relational databases and spreadsheets to analyze them, such an amount of data yields the impossibility to study large samples of data. It gets worse if those companies core workforce is made of domain experts for which

1. https://www.star.fr/

computers are tools and services providers only.

For the case of the bus network of Rennes, it generates nearly 2GB of data per month for the sole AVL system, and more than 10GB if we consider all the stakeholders of the bus network information system. Thus, for the domain experts of Keolis Rennes who are provided low/middle end computers, with spreadsheets software and no access to big data solutions, the analysis of large samples of data is quite a challenge when it is not merely out of reach.

1.1.2 Motivation

The current situation in Keolis Rennes is the following:

— The bus network information system generates flows of erroneous data that has to be governed. That is to say that data management systems dedicated to data cleansing, enriching and centralizing have to be built while keeping them out of the existing information system in order not to have any side effect on it.
— The planning of the bus network day to day life is based on domain experts knowledge, which are not able yet to make use of the massive amount of data they have under their feet to precisely build bus scheduling and bus lines travel time scheme.
— The strategic decisions such as the creation of new bus lines ex-nihilo are made upon domain experts knowledge without reliable ways to predict e.g. speed, travel time, etc.
— There are no integrated tool that could help the operators making decisions based on the historical knowledge that data offers.

It appears that a tool that integrates data management, data analysis, and prospective/predictive models would be a great breakthrough for Keolis Rennes, making the use of its massive data possible by its operators.

In order to understand why those problem are yet to be handled, this chapter explores the state of the art of Urban Public Transportation Networks design, optimization, information systems but also simulation and prediction tools. In addition, we take a look at the state of the art of model-driven / data-centric models in software engineering, domain that offers promising ways to produce new software tooling for Urban Public Transportation Networks.

Acronym	Full name
UPTN	Urban Public Transportation Network
UBN	Urban Bus Networks
PTIS	Public Transportation Information Systems
IPTS	Intelligent Public Transportation Systems
UTNDP	Urban Transportation Network Design Problem
RNDP	Road Network Design Problem
PTNDSP	Public Transit Network Design and Scheduling Problem
AVL	Automatic Vehicle Location system
SCD	Smart Card Data system
TIS	Travelers Information Systems
BRT	Bus Rapid Transit

Table 1.1 – Public transportation network glossary

1.2 Urban Public Transportation Networks, their information system and related data

Urban Public Transportation Networks (UPTN), amongst other complex transportation networks are more and more integrated within smart cities. Hence they can be considered as both large scale probes that generates data about what happens in a city (people traveling, traffic congestion, air quality, ...), and levers on which one can act to evaluate the impact of, e.g the rerouting of a bus line on passengers flow and/or traffic jams, etc. This section is dedicated to existing work leveraging public transportation networks in all their aspects, from public transportation networks structure and optimization, Public Transportation Information Systems (PTIS) and Automatic Vehicle Location Systems (AVLS) to data issues.

1.2.1 Urban Public Transportation Networks

Urban Public transportation networks are complex socio-technical systems [51, 58] that involve humans (drivers / operators, passengers, pedestrians,...), economic entities (cities or companies), laws and specific rules (speed limitations, fees, schedule, ...), hardware (vehicles, roads / public infrastructure) and software (fleet management, Automatic Vehicle Location systems (AVL)[2, 56]) Entities that manage this kind of network have to tackle the Urban Transportation Network Design Problem (UTNDP) as presented in the reviewing works of [20, 34]. This issue is two sided. First, the Road Network Design

Problem (RNDP) which consists in optimizing the creation of new streets and/or the expansion of existing streets. Second, the Public Transit Network Design and Scheduling Problem (PTNDSP), that aims at offering the best trade off between performance (high number of passengers per vehicle, low number of vehicles, fast traveling) and costs [8]. Hence, decisions included in UTNDP can be reduced to 3 major domains:

— Strategic, which contains the long term decisions such as the building of new streets or new bus routes.

— Tactical, which consists of the optimization of the already existing infrastructure and bus fleet, e.g determining whether electrical buses should be used on a specific bus line regarding its overall steepness, or turning a road portion into an exclusive bus lane.

— Operational, which is the ensemble of short terms decisions such as the scheduling of traffic lights, roadworks, etc...

The work of [11] describes the Bus Network Design Problem, which is directly related to UTNDP, more specifically to PTNDSP. Urban Bus Networks (UBN), as urban public transportation networks that are intertwined into their host city's traffic and public infrastructure, are particularly sensitive to internal and external factors such as traffic load, bicycles, road infrastructure, transportation demand, fleet size, weather, payment system, etc [55, 32, 35, 23, 44, 65, 15]. This implies that in addition to UTNDP, the building and transformation of all or parts of most of such a network ask for solid theories and domain expert not to reduce its performance and attractiveness [11].

Urban Transport Networks optimization

The optimizing of an UPTN requires to tackle the Urban Transportation Network Design Problem as stated by [20]. This problem can be apprehended in several ways amongst the following: Greenhouse gaz emission and environmental impact minimization, station spacing optimization, passenger flow optimization, cost optimization, travel time optimization, fleet optimization, transit optimization, etc. Its major goal is to provide the best possible service with a minimized service cost (economical and environmental) and travel time and a maximized number of passengers per vehicle.[11, 20]

Weng et.al [71] proposed a visual analytics approach named BNVA (Bus Network VisualAnalytics system). This work aims at providing domain experts a tool that helps to find candidates for either new bus routes or inefficient bus routes (that they also identify thanks to BNVA). They used 3 different datasets, the first one being the bus

stops dataset which contains bus stops meta data, the second one the bus routes dataset, consisting of bus stops sequences and the last one the trips dataset made out of travelers card validations bound to bus routes. Those datasets feed a graph model that generates candidates routes based on an optimum defined as a given ratio between passenger flow, service cost and short transit distances from stop to stop. This ratio is computed by a Pareto-optimal searching algorithm based on a Monte-Carlo search tree. This work is a first step towards centralized data systems to tackle UTNDP thanks to its visual tool and the using of real data at the inter-station granularity. However it misses data inputs such as traffic, infrastructure, fuel consumption, etc. Also its optimization algorithm is Pareto based, which is probably less suitable than using machine learning with real data because Pareto decisions are not directly correlated to what happens in the real world. i.e some silent biases can be manually introduced while tweaking the Pareto algorithm.

Markellos et.al [42] worked on a Multi-layer Perceptron (MLP) Neural Network (NN) approach to evaluate the efficiency of Public Transportation Networks. They focused on the maximization of miles per year with a restricted amount of drivers and vehicles. The efficiency definition in this work is probably out of date as of 2021 as long as climate change and social issues imply to enlarge this definition to include air quality, servicing low population areas, extending service hours, etc.

Tirachini et.al [66, 67, 65] worked on the estimation of bus travel time and the impact of fare payment technology and bus floor level in urban bus services. Their findings is that card payment systems seems to be always better than cash payment systems, but also that off-board payment systems are best suited when one needs to operate fast buses, regardless from the existence of dedicated bus lanes. They also consider infrastructure impact on bus network performance in dedicated bus corridors and studied the impact of station spacing, running speed, bus capacity and frequency, congestion amongst buses and infrastructure cost per kilometer on the demand regarding the fare payment system. The goal is to assess which feature to act on depending on the demand and maximum operational cost while keeping a low travel time. Their models are based on empirical studies with manually collected data on a sub portion of the Sydney's bus network. This system lacks large scale automatic data gathering.

Yu et.al [73] proposed an Ant colony approach for the transit optimization problem, which consists of building a bus network in which transfers of passengers between bus lines is minimal. This approach is tackling only part of the UTNDP, it would be interesting to extend this experiment by adding more variables like air quality, servicing low population

areas, extending service hours, etc.

Gormez et al [26] worked on an emerging problem which is the optimization of the charging cost of a fleet of electrical buses deployed over a bus network. They proposed an algorithm that takes in account physics (vehicle mechanics, elevation, wind/air condition,batteries degradation cycle and capacity) in order to optimize energy consumption, charging time and overall cost but also to minimize the batteries degradation over time and yet keep a satisfying operational speed over the bus network. However this optimization works better if charging is possible at each bus stop, allowing the lightening of batteries and spreading of charging loads. This work is one of the few which propose an approach to optimize electrical buses usage in a bus network. However, depending on the city bus policy and financial means, it might be necessary to adjust their proposal in order to make it fit any situation, typically cities in which charging at each bus stop is not feasible. Also, the optimization process asks for data that can be tricky to obtain at a large scale in time and space amongst road grade, passengers, road conditions, wind velocity, and traffic regulation.

Bel and Host [7] worked on the evaluation of the impact of Bus Rapid Transit (BRT) on air pollution in the city of Mexico. They found that BRT systems seem to help the enhancement of air quality. Jointly, Sun et al. [61] led an empirical study in china cities to measure the variation of air quality with the providing of more buses in bus networks. They found that adding more buses (resp. services) helps the enhancement of air quality by a significant margin whilst it tends to be underestimated by cities. These works are amongst the few that tries to tackle the air quality problem related to urban bus networks. More data should be collected and used in decision models to assess their work in other places in the world, for example by placing air quality sensors on buses as proposed by [37].

As we can see, these works tackle different parts of the same problem: the UTNDP. Yet, none of them offered a way to consider this problem as a whole with a direct and on-line connection to operational data.

1.2.2 Urban Public transportation networks' information systems, and data

In the previous section, we exposed the basics of urban public transportation networks and their optimization. Here we discuss their operational aspects. As cities get bigger and more crowded, UPTN get wider and more complex too. Thereby, it is of uttermost

importance to use computerized means to ensure a correct management of those networks. Digital tools that, consequently, generate loads of data useful for broad studies of UPTNs are then deployed.

PTIS

Automatic Vehicles Location systems (AVLS) [56], Smart Card Systems (SCD) [53] and Travelers Information Systems (TIS) [1] are computerized management systems used as follows:

— AVLs help the handling of a public transportation network's vehicles fleet by gathering data about vehicles positions, speeds, dwell times at stops, and other various metrics or meta-data. This data can then be sent in real-time via wireless technology (GSM or other large scale wireless networks) to the data center of the network manager. It is also possibly kept on board until the vehicles return to the depot and then upload the data they acquired using wireless technology (e.g WiFi) to the data centers [2, 56].

— SCDs collect data about the usage of the network by gathering travelers card validation data, in order to measure the ridership of the different lines of the network and acquire a better knowledge on how the network is used. Eventually this data can help to optimize the offer and services regarding the actual demand. [53]

— TISs provide information to travelers all over the network, e.g the arrival of the next vehicle at a station, or even planning a trip via a mobile application [36].

In standard situations, AVLs, and TIS are embedded and working alongside each other within a Public Transportation Information System (PTIS), also called Intelligent Public Transportation Systems (IPTS) as explained by Elkosantini and Darmoul [18]. However SCD and other tools such as fuel management systems, geographical information systems or scheduling systems can be totally independent from the PTIS, yielding heterogeneity.

Data errors and data quality issues

PTIS and analogous systems are made out of hundreds of sensors and communication systems deployed all over the UPTN. Hence, such a quantity of various and connected hardware is a heavy task for maintainers. Also, the bigger the system is, the higher the system is prone to errors. The large amount of different pieces of hardware combined with the use of wireless technologies for data transfer result in a high risk of errors in data. The sensors can be miscalibrated, faulty, or even out of order, yielding erroneous data or

merely none. In the other hand, wireless data transfer between e.g. vehicles and servers can be failing and part of the data can be lost if there is no fault tolerant system that prevents the data from being lost permanently during the transferring. Hence, there is a need to improve the quality of this very noisy data in order to enhance the precision of the machine learning algorithms that make use of it. Indeed, the noise tolerance of machine learning algorithms might be insufficient in this very context and the use of non-cleansed data can result in poor predictive models.

Smith et al. [60] proposed a set of statistical tools to handle missing data in transportation management systems. Their work shows that it is possible to infer missing data with a satisfying error rate, as long as the correct heuristic is used for this purpose.

Ma and Chen [41] explored the data quality of smart card and GPS systems. They claim that it is needed to use redundant data when it is available to recover or correct missing values and that any useless field should be removed from the smart card data sets. Also, they claim that the error identification is made harder by the complexity of transportation systems that are often made of software blocks managed by different manufacturers and stakeholders. They say that this should urge the data manager to work on the data consistency before data collection to make sure that all the production data is based on the same units, definitions and same accuracy level.

Robinson et al. [57] studied methods to improve data quality of smart card systems. They identify different sources of error in 4 investigation domains that are software, hardware, data and user. They proposed a method to identify boarding and unboarding data errors and faulty data supplies. They claim that taking care of smart card data quality can reduce costs and enhance the service quality offered by the transportation network.

1.2.3 Conclusion

In this section, we explored the public transportation networks state of the art. The observations we made yielded several points. First, as complex socio-technical systems, the optimization, planning and modification of PTNs are difficult tasks for which there are no simple or perfect solutions. Second, their operational systems, including Information Systems and management systems tend to be highly heterogeneous, asking for a high domain knowledge and competency. Third, the data generated by PTIS tend to be full of errors. In the same way external public sources one could use can be difficult to manage or obtain at a larger scale than a portion of road or a single bus line. This makes the usage of the data of PTIS to address the UTNDP a heavy task for which there is no out

of the box toolset yet.

1.3 Predictive analytics of bus networks

A major issue of public bus network is the prediction of either the existence of a future section of a network (e.g a path between two bus stops that should be considered by the operator) or of an event and its properties (e.g the speed of a bus on a given line with a given timestamp). In this section we explore the state of the art of analytical, simulation and machine learning based models for public transportation networks.

1.3.1 Analytical models

There only are few analytical models for public bus networks: Fernandez and Valenzuela [23] proposed an efficient analytical model to predict bus commercial speed anywhere on a bus network, for any kind of bus line. They upgraded a state of the art function based on exponential decay to take in account more influencing parameters such as dwell time at bus stop, passenger density or even time periods or bus technology. Their model must be precisely parameterized by tweaking weights in order to produce satisfying result regarding the reality. Thus, one who wants to use this model would have to gather a lot of data and domain knowledge in order to obtain satisfying results. In the same way, Valencia and Fernandez [69] developed a similar approach dedicated to bus corridors (bus lanes). Their work presents similar characteristics, hence pros and cons, as the model described before.

Analytical models are built by and for specific issues. In these particular contexts they can be very powerful when adequately tweaked. If one wants, for example, to predict the travel time of a bus line for which all the properties are known in advance using a perfectly configured model, these models can provide an interesting answer. The other side of the coin is that these models are very static, hence non applicable to others issues without a total rethinking of their behavior. Moreover each model is tweaked to fit a specific situation, making their portability to other instances of the same problem probably not advised.

1.3.2 Simulations

Simulations are dynamic models that aim at providing estimations obtained from a simulated environment. The closer the environment is to the actual environment, the better the estimations are expected to be. This kind of tool is usually dedicated to a specific task at a given granularity. Indeed, simulations are usually classified in 3 different categories [5]:

- Macroscopic: Simulation of a whole city (or even more), emulating traffic flow dynamics (inspired by fluids physics)
- Mesoscopic: Simulation of a district, involving explicit treatments of intersections
- Microscopic: Simulation of a crossroad or a few roads. Multi-agent based with complex rules and interactions between agents.

In Public Transportation Networks, because macro and meso simulations are not as precise and fine-grained as microsimulation, the latter is preferred [10]. It is used for strategic purposes such as the placing of bus stops before or after a crossroad, the width of the roads, etc.

Fellendorf [22] proposed VISSIM, a microscopic simulation tool to evaluate actuated signal control including bus priority. His work proposes to build simulations to assess whether a traffic light equipped crossroad is properly configured for bus priority or not. As a microscopic simulation, it is designed for small scale evaluation, a few crossroads maximum, also an important amount of time and effort has to be dedicated to configuration [74].

Esser and Schreckenberg [19] worked on an urban traffic simulation based on cellular automaton. They built a simulation in which cellular automaton are used to describe roads as arrays of cells connected to each other. A free portion of road is represented by en empty cell while a vehicle occupies one or more cells depending on its size (e.g one cell for a car, 3 for a truck, etc.). This simulation is traffic focused and aims at providing a step by step simulation behaving in accordance with traffic rules defined in the system. It is able to gather data about vehicle flows, travel time at different point of view (e.g vehicle $\rightarrow$ road $\rightarrow$ district).

This simulation is certainly useful for network scale simulations, as they managed to simulate days of traffic on a virtual network totaling 165 km of road over 22059 cells in 20 minutes on a computer of the late 90's, although the creation of the model can be very time consuming (each road and specific properties has to be manually built into the system). In the end with modern hardware the computing cost of a simulation containing

hundreds of thousands of cells on which thousands of vehicles are represented is certainly not a problem. However the design of the simulation might be the bottleneck here.

Cats et al. [10] worked on a mesoscopic model for bus public transportation. Their system is based on Mezzo, an object-oriented and event-based mesoscopic traffic simulation model. Their tool is used to simulate operational bus lines and control their behavior (headway, delay, etc). They manage to simulate the behavior of bus line 51 in Tel Aviv, Israel. Their model ran in 10 seconds on a low end computing device as of 2010. The results of the experiment show that the model is quite capable for running time prediction, but not as good for headway prediction. Yet, they claim their model can be extended to simulate other issues. However this model is focused on single bus lanes and does not simulate the whole bus network at once, even if this might be possible regarding the model's low computing cost, provided it is feasible to implement multiple bus lines at once.

Matsumoto et al [43] worked on bus line optimization using multi-agent simulation model and origin-destination data. Their model can be considered has mesoscopic: It represents a 65.63 km^2 portion of the city of Toyama, Japan. However, it is nearly the surface of the whole city and their model is multi-agent based so it is actually more like a microscopic model applied at a meso/macro scale. In terms of performance, their model performed quite well with a 50 hours running time for 181'000 agents, which is quite acceptable knowing the complexity of the simulation. Eventually, they built an application example for community buses in Imizu, Japan. It appears that their model is able to propose new optimized routes regarding the Origin-Destination needs and behaviors of agents in the system. However, they use mathematical workarounds in order to reduce computing time and candidates selection, which possibly obliterates valuable candidates. Moreover, this model needs a lot of modeling and configuration to fit each environment and situation.

Barcelo et al [5] tried to tackle the macro/meso/micro junction problem which consists on building a model capable of doing simulations at macro, meso and micro scale. Their work lies on AIMSUN NG simulator which is a traffic simulator. They built a common database that allows the transitioning between the 3 levels of simulation, solving the consistent network representation problem. It appears that the system is quite consistent when it comes to calculate shortest path using either micro or meso simulation, which indicates that the database is well built. However, this system lacks real world data: only voluntarily generated jams, intersections and traffic signals are taken into account in the core of the model.

1.3.3 Machine learning approaches

Machine learning models are broadly used for bus networks problems modelling. Among other pros, machine learning models hardly ask for heavy configuration and/or wide domain knowledge. On the other hand, a major caveat of ML is its dependency to data. Consequently, and knowing that real world data tends to be dirty [49, 60], data quality (truth, consistency, accuracy, etc.) must be considered for these models.

Data issues in machine learning applied to public transportation networks

There are only a few studies on data quality in transportation. Brian L. Smith et al. [60] proposed a set of statistical tools to handle missing data in transportation management systems. Their work shows that it is possible to infer missing data with a satisfying error rate, as long as the correct heuristic is used for this purpose. Their work is part of the world of global transport, which is close to urban public transportation, yet there might be differences in the approaches depending on what kind of data is available and the desired outcome.

Ma and Chen [41] explored the data quality of smart card and GPS systems. They claim that it is needed to use redundant data when it is available to recover or correct missing values and that any useless field should be removed from the smart card data sets. Also, they claim that the error identification is made harder by the complexity of transportation systems that are often made of many manufacturers and stakeholders. They say that this should urge the data manager to work on the data consistency before data collection to make sure that all the production data is based on the same units, definitions and same accuracy level.

Robinson et al. [57] studied methods to improve data quality of smart card systems. They identify different sources of error in four investigation domains that are software, hardware, data and user. They proposed a method to identify boarding and unboarding data errors and faulty data supplies. They claim that taking care of smart card data quality can reduce costs and enhance the service quality offered by the transportation network.

In the light of their arguments, it appears that the IT systems of transport networks are rich and complex. However, the data that emanates from them should be viewed with caution as the presence of errors within them is prevalent, making any work based on them inherently delicate.

Machine learning based models

Machine learning models are mostly used for travel time prediction in the literature: Altinkaya and Zontul [3] reviewed the computational models used in Urban Bus Arrival Time Prediction as of 2013. Their work covers the use of historical data and statistical models, time-series, regressions, neural networks and hybrids models. They explain that the existence of that many models in this field of research is due to the fact that predicting travel time of buses is a complex task for which no model in particular has proven its superiority yet. However, they claim that hybrid models (e.g combine neural networks with Kalman filters) are on the roll. Moreover, they add that predictions might be better if one splits the datasets into sub-datasets in which data represents similar conditions, restricting the models prediction scope.

Some other interesting machine learning must be noticed too: Zaki et al. [75] worked on a hybrid neural network and Kalman filter model to predict online bus arrival time. Their method is a hybrid scheme that combines a neural network (NN) that infers decision rules from historical data with Kalman filter (KF) that fuses prediction calculations with current GPS measurements of buses. They feed the neural network with the following features: Day, Direction, Stations, days category, weather, average speed and traffic status. They conducted an experiment to test their approach by creating virtual bus line and a set of random data generated through Matlab dedicated process. Their results show that this model can perform reasonably well with simulated data. However, the discussion lacks validation metrics and a real world application to assess the model portability. Napiah and Kamaruddin [48] worked on Auto-regressive Integrated Moving Average (ARIMA) models to predict bus travel time. They tested their model in a case study over Ipoh-Lumut corridor in Malaysia, that link the Ipoh city to the port of Lumut. The bus line they studied was 83 km long, hence it can be considered as suburban / middle range bus line. They collected data through a survey, gathering information for 48 trips over a year. They obtained satisfying results with their models while predicting full line travel time, with Mean Absolute Relative Error (MARE) and Mean Absolute Percentage Predicting Error (MAPPE) respectively below 8 minutes and 6.5%. Those results are quite good if we take in account the length and travel time of the bus line. Their work showed that ARIMA models perform reasonably well when one does not have detailed data of affecting factors such as traffic, weather, etc. However, their result analysis lack other validation measures that emphasize outliers such as Root Mean Squared Error (RMSE), hence it is hard to tell if the model performs steadily or if it can predict aberrant travel times. Cristobal et

al. [16] proposed a bus travel time prediction model based on profile similarity. They used a k-medoids clustering algorithm to build historical travel time profiles using historical data. Then they compare the profile of the bus they want to predict travel time with the built profiles set to obtain a predicted travel time. One major advantage of this model is that it does not require recent travel time data to provide a prediction, which makes it suitable for onboard prediction. Also their result showed that their model has a 13% Mean Absolute Percentage Error (MAPE) margin, which is on par with neural networks methods they compete with. However this margin of error can be important depending on the length and total travel time of bus lines. Also, other validation methods like RMSE should be computed to assert whether the model is stable or not. Mendes-Moreira and Barachi [45] imagined a prediction model for networks by predicting sub parts of the networks and re-conciliate the aggregated predictions of the sub-parts with the path they are part of. They do this using a method they called Reconciliation For Regression (R4R) by weighing every sub-predictions using a constraint least square algorithm. Their results show that they reach state of the art performance for bus travel time prediction. However, it is unclear on how far from the reality their model perform without MAPE. One could also raise the following statement: the added complexity of R4R is questionable because it shows that it seems to never offer a better improvement than 3% in prediction precision when compared to other models, including simple ones such as Multivariate Linear Regression (MLR).

1.3.4 Conclusion

In this section, we exposed the state of the art of prediction tools and models for UPTN, specifically bus networks. We showed that there are plenty of methods that are being explored by researchers, yet none of them has been able to make a significant difference amongst them. Also, we observed that most of those models are designed for either a specific environment (modeled by and for a given city / bus line) or a given target (travel time, optimized bus path, ...) It seems that if it is probably out of reach to produce a prediction model that definitely surpasses all the others, it could be possible to produce a set of methods that offer a flexible prediction tool set for bus networks operators. As of today, those predictive models are used independently from data management and analysis systems. Moreover, they seem to be designed for either mono-dimensional predictions (e.g. whole bus line travel time, inter-station time of arrival, etc.) and/or single targeted predictions (travel time, speed, arrival time, etc.). Thereby, there might be a need for

integrated multi dimensional and multi target predictive models.

1.4 Software Engineering models

Software engineering techniques can be leveraged to tackle UPTNs design problems. Thanks to the many data UPTN generate, the creation of virtual representations is desirable for, e.g. spatio-temporal data analysis, and even predictions using predictive models. To do so, there are some existing works that can help clearing the path.

Combemale et al. [13] proposed the conceptual models and data (MODA) framework. They aim at providing a reference for model-driven and data-driven modelling issues. Their work proposes a data-centric and model-driven approach to integrate heterogeneous models and data into a single framework for the entire life-cyle of socio-technical systems. Depending on the structure of the considered socio-technical system, its whole life-cycle might be impossible to integrate. Actually, the proposed framework is pretty flexible, thereby one who wants to produce an instance of it for a specific need can ignore parts of the model that are optional, such as descriptive and predictive models. Moreover this work proposes a solid way to evaluate and build a model that gathers internal/external data into a single framework that would manage whole or part of the life-cycle of the system depending on the needs. Actual applications of this framework have yet to be built in order to assess its portability to real world issues.

Hartmann et al. [29, 27] worked on temporal graphs to analyze data in spatio-temporal dimensions, which embed historical analysis and predictions. Their model, Greycat, is a scalable graph-oriented data model that allows the building and analysis of multiple parallel worlds (forks of graphs) in a single framework. Their model is meant to be used for fast evolving networks such as smart-grids, cyber-physical systems or IoT systems that yield a lot of data and that can physically evolve quickly. The strengths of their model are the following: their model is totally and quickly scalable with reasonable resource needs; they embedded machine learning seamlessly in order to predict events on the graph and even build what-if scenarii by forking graphs, yielding new instances with inherited properties; moreover, they took a look at the interest of applying the "divide and conquer" paradigm in order to make predictions over the smaller parts of the graph (Nodes). Their findings is that for complex data models that are made of an aggregation of smaller parts, machine learning models that are trained using fine-grained data outperforms models trained with coarse-grained data (i.e parts or whole graph). Greycat has been successfully

tested on a smart-grid application in Luxembourg, and is an interesting way of thinking the interactions between data and models at large scale on evolving environments. Unfortunately, Greycat was not fully available when this thesis was written, and probably too tricky to integrate in a research environment for yet non-built applications such as urban public transportation networks. However, one can further validate the fine-grained and temporal-graph approaches for socio-technical systems by applying them to use cases that differs from smart-grids.

Another meta-modelling work of Hartmann et al. [28] tackled the issue of meta-learning for machine learning. Meta-learning for a given prediction task consists of auto-selecting the best predictive model, auto-extracting the features from the training dataset and finding the best configuration for the selected model. This issue is both complex and costly because as of today there are multiple machine learning algorithms with various cost of use and predicting performance. Meta-learning enabled meta-models would be a way to reduce the endeavour data scientists have to furnish when it comes to build predictive models over specific datasets, but this would probably ask for a high computational cost.

Bordeleau et al. [9] suggested that models at runtime are key for implementing digital twins. From a Model-Driven Engineering (MDE) point of view, they are virtual representations of Cyber Physical Systems (CPS). Digital Twins have to be tightly bound with the data that is generated by the CPS in order to be a consistent virtual representation of this very CPS. However, this raises challenges such as bi-directional synchronization with the actual system, the management of heterogeneous models, and the collaborative development throughout the system life-cycle. Finally, their work highlighted open research challenges that one should consider when creating any digital twin (meta) model.

Kirchof et al. [38] worked on the inter-connectivity of Digital Twins, their related Information System and the Cyber Physical System they are the virtual image of. They claim that a significant part of digital twins development is due to the creation of interfaces that are in charge of the data exchanges between the digital twin and its relative CPS. Their work could be used to enhance the inter-connectivity of any digital twin and the CPS they are the virtual representation.

1.4.1 Conclusion

In recent years, data and software engineering models built for networks (Smart-Grids, Cyber-Physical Systems, Socio-Technical Systems,...) are emerging. Researchers have started to imagine models that can fit different situations from smart-grids to Crisis

management systems. Those models propose new seamless approaches, with embed data exploration, prediction and prospective. However real-world applications are technically difficult to build because numerous issues have to be handled: data gathering, model adaptation, software development, non-regression regarding the existing systems, etc. Hence, a framework that allows seamless integration of data along with spatio-temporal data analysis tools into models has yet to be imagined.

1.5 Overall conclusion

In this chapter we studied the state of the art of Urban Public Transportation Networks (UPTN) and their existing issues. We also exposed state of the art tools for bus networks optimization and prediction. Finally we discussed the existing software engineering methods and frameworks that exist to address data-driven issues similar to those we try to tackle in this thesis.

It appears that UPTN optimization issues are both complex and many-fold: from structural optimizations (roads, bus stop spacing, ...), to operational optimizations (schedule, fleet management, headway optimization, ...), there are plenty of problems to address with as many different tools for that. Moreover, for both optimization and prediction tasks, there are no best-suited tools yet. Thus, human intervention is still necessary to properly tackle those issues.

Feature	**Simulations**	**Analytics**	**Machine Learning**
Needs domain knowledge	✓	✓	∼
Heterogeneous data sources	✗	✗	✓
Uses real data	✗	✗	∼
Spatio-temporal	✓	✗	∼
Data analysis	∼	✗	✗
Multi-targets predictions	✓	✗	∼
Evolutive	✗	∼	✗
Scalable	✗	✓	∼
Portable	✗	✗	∼

Table 1.2 – The thesis positioning in the predictive analytics of bus networks state of the art

Table 1.2 shows the current predictive analytics of bus networks approaches in the state of the art. It exposes eight features for each of the different approaches :

— **Needs domain knowledge** Describes the need in knowledge related to the approach itself and UPTN to design, feed and use a model.
— **Heterogeneous data sources** Describes the capabilities of the approach to make use of heterogeneous data sources (i.e different databases, different formats, etc.).
— **Uses real data** Shows the ability of the approach to make use of data extracted from the real world (AVL, smart card data, etc.).
— **Spatio-temporal** Represents the ability of the approach to be used for any period in time and any scale on the bus network, without requiring reconfiguration.
— **Data analysis** Represents the capabilities of the approach to analyse the data it uses, contains or represents.
— **Multi-targets predictions** Corresponds to the ability of the approach to predict multiple targets (e.g speed, travel time, etc.) and configuring prediction for new targets at run time.
— **Evolutive** Represents the way the approach can evolve through, e.g the adding of data sources, predictive models, etc.
— **Scalable** Represents the way the approach can keep on working when a lot more data, predictive models, etc. are added to it, without significantly deteriorating the performance.
— **Portable** Represents the ability of the approach to be used for issues different than UPTN.

The different symbols ✓, ∼ and ✗used in Table 1.2 have different meanings:

— ✓: Feature totally supported by the approach.
— ∼: Feature partially supported by the approach. i.e an endeavour is needed to support the feature.
— ✗: Feature unsupported by the approach. i.e an intense endeavour is required to support the feature, if the approach design authorizes the implementing of the feature.

If we summarize Table 1.2, we can say the following things:

— Simulations approaches [5, 10, 22, 74, 19, 10, 43, 5] need a lot of domain knowledge. The building of a simulation is a complex task that requires to virtualize the rules of the real world. By design, a simulation has to be configured for a specific target (in a set of available targets), in both space and time aspects and run. This implies that the use of heterogeneous data or real world data to run the simulation is nearly impossible. However, in some cases, it might be possible to analyze the data the

simulation produces, but it is probably not easy to merge this data with exogenous data for broader analysis. Simulations are complicated to evolve, because of the inherent complexity of the models, hence scalability has to be integrated by design if one wants the model to be scalable. Finally simulations are dedicated to a specific task, hence it is quite impossible to use it for any other unrelated issue.

— Analytics approaches [23, 69] need a lot of domain knowledge because the designing of analytical model requires to master and understand what one wants to represent. However, the model itself is not able to use heterogeneous data sources or real data. Indeed, it only uses local variables, hence it does not integrate data analysis tools nor spatio-temporal abilities. Moreover, analytical models are designed for a single target (speed, travel time, etc.) and cannot be used to predict multiple targets, by design. On the other hand, as it is very light (just a simple equation), it is totally scalable. Finally, this approach yields models that are evolutive, but evolution of the models requires an important effort.

— Machine learning approaches [3, 75, 48, 16, 45] can be used without domain knowledge by, e.g feeding a CSV to a predictive model and hope it will work. That is to say, the domain knowledge needed here is more related to machine learning approaches themselves than the domain one applies those models to. On the other hand, one can feed machine learning models with heterogenous datasets that come from the real world or that are artificially built. However, the predicting of multiple targets over space and time with machine learning approaches requires those models to be designed for this. i.e, wrapped in a component that manages the spatio-temporal stuff and multi-targets with e.g. a set of different predictive models for the different targets. Moreover, machine learning approaches are not evolutive, in the sense that the evolution of machine learning models consists of depreciating the current model and training a new one with the new data. In the same manner, the portability of machine learning approaches is not "out-of-the-box" as long as the gathering of new data is required along with feature selection and re-configuration of the models. Finally, machine learning approaches are not designed for data analysis and their scalability depends on the design of the models, which sometimes are (e.g.: random forests), and sometimes are not (e.g. linear regression).

With the spreading of cities areas, bus networks are prone to yield more and more data through time. Hence, data-driven analysis and decision models are probably intended to be

created and deployed in the upcoming years. in Section 1.2, we saw that the management and optimization of UPTN is composed of complex issues that cannot be handled without proper domain knowledge and data. Section 1.3 gave us hints about the use of predictive models for UPTN problems such as travel time prediction, highlighting the fact that the use of data generated by PTIS can help to reduce the level of domain-knowledge and expertise needed to make good predictions. Finally, Section 1.4 tells us that the integration of predictive models in models at runtime research is still in its infancy. Thereby, we were able to formulate the statement that this thesis defends:

Main statement

Integrated tools for data analysis and decisions making has become a requirement to cope with smart cities development. To fulfill this requirement, the designing of a system that helps for the UTNDP, relying on bus network data (AVL, smart card, ...) and exogenous data (road infrastructure, traffic, ...) through the integration of data analysis and predictive models is now possible. Our thesis is that this opens new path towards better integrated tools for decision making in the context of spatio-temporal models.

In this thesis, we propose four contributions, which support all the features of Table 1.2 when assembled. We propose a software engineered solution towards data centric decision models for Urban Public Transportation Networks, with a real-world application on the bus network of the city of Rennes, France. The first contribution addresses the data quality issues in UPTN. The second one assesses the impact of exogenous factors on bus speed using large-scale real world data. The third one proposes a fine-grained prediction approach using real world data to predict bus speed. The fourth and last one proposes a framework that provides spatio-temporal data analysis and prediction tools for bus networks operators.

Chapter 2

Data quality and integration

2.1 Introduction

Current Public Transportation Information Systems (PTIS) produce a huge amount of heterogeneous data on a daily or even real-time basis. For example, smartcard systems, on-board bus units (AVLs [1]), schedule, referential, and real time radio bus monitoring systems are a gold mine to manage the bus network, useful to understand how it works and what to act on to improve it. For example, the average trip travel time, the schedule, the amount of km per year, the average bus stop spacing, etc. can be considered as inputs that, when properly processed, help the creation of service quality indicators for bus networks [23, 15, 14].

Many organizations are thus interested in trying to apply machine learning techniques to a wide part of their PTIS. These could be used to predict local and global behavior of the network, such as arrival time for each station [3], ridership [17] or even to elaborate what-if scenarios when considering road works or network enhancements.

However, the results of any prediction task may be inaccurate, due to omissions and errors in the input data set. PTIS are a typical example of error prone systems, for numerous reasons [41, 57]. PTIS are made of many independent software and hardware that (try to) communicate using different, ad hoc protocols. Bus fleet are composed of hundreds of buses that embed numerous sensors and wireless communication systems that can fail. Risks of data loss come from the wireless communication system, the embedded hardware on buses and the number of buses in the fleet that have a variety of hardware to maintain. That variety increases the risk of errors, but also makes it difficult to identify the exact source of the error when it happens. Overall, cleansing this heterogeneous data is costly [70].

In this chapter, we investigate the question of data cleansing for a typical machine learning task in PTIS and a key issue for the Keolis company: predicting bus commercial

1. Automatic Vehicle Location

speed.

Commercial speed definition

The commercial speed is the average travel speed of public transport vehicles between one origin and one destination stop, including any delay encountered during the trip [23].

We ground our analysis on the PTIS of Keolis Rennes that exploits the bus network of the city of Rennes, France, yielding 18GB of data per year.

In this specific domain, little is known [41, 60] about the amount of effort one has to make to obtain accurate predictions. We consider a global data cleansing strategy with various levels of quality, ranging from easy/cheap ones to computational intensive ones, and observe their impact on prediction quality. We demonstrate experimentally that cleansing is mandatory, but also that a complete alignment/completion of all the available data sets, involving data synchronization and complex joins, is of little interest. More precisely, our contributions are the following:

— We define a datalake gathering all the information from the operation PTIS of Keolis, for a middle-size metropolis, Rennes, in France (730k inhabitants);
— We identify the errors in this datalake;
— We present a dedicated data cleansing strategy capable of yielding different levels of data quality;
— We evaluate the resulting commercial speed prediction precision on several, real size data sets.

The rest of this chapter is organized as follows. In Section 2, we present the information system and the data sets we got from Keolis Rennes. In Section 3 we explain the various kinds of transformation and cleansing we can apply on the data. In Section 4 we compare the results of commercial speed predictions on two of our data sets with different level of data quality. In Section 5 we discuss the results and expose our threats to validity. Section 6 presents the related work before we conclude.

2.2 Data sources

Taking the Keolis Rennes example, we gathered data on a 6 months period between early July 2018 and late January 2019. The data collected are based on the bus network

which contains 116 bus lines.

We had access to three data sets from the PTIS's AVL system of Keolis Rennes:

1. The referential and schedule: REF;
2. The on-board central units data set: OCU;
3. The radio real-time monitoring data set: RT.

The referential data set (REF) contains topological information over time. It describes the network as an oriented graph in which vertices are bus stops and edges are inter-stations. An inter-station is a path between two bus stops. Bus lines are defined as paths through inter-stations within the network graph. The schedule contains temporal information about the bus services, e.g scheduled time of arrival at bus stops over time. The referential and schedule volume of data per year is usually around 1 GB.

The on-board central units data set (OCU) contains *a posteriori* data. It is the record of bus trips structured as a table in which each row is a record of metrics of a bus serving an inter-station (or inter-stop), at a given instant. On-board sensors provide the data of this data set, hence it is expected to be accurate, but for reasons explained above, it is also incomplete and full of errors. It provides meta data such as bus, line, schedule information and metrics like commercial speed, travel time, traveled distance, etc. Most sensitive among these for contractual reasons, the commercial speed is the speed of the bus between two points, including travel time and dwell time at the origin point [23]. The OCU data set is made of 35 fields and represents around 10GB of data per year with daily file sizes varying between 9 to 40 MB. Some of the missing data from the OCU data set however could be inferred from the radio real-time monitoring data set (that collects real time data by radio). Indeed, RT contains 13% more recordings than OCU.

The radio real-time data set (RT) is an historized data set. It is the record of real-time data traveling through TETRA [2]: it contains data that is somehow a duplicate of the data from the on-board units data set. However since it is a monitoring data set with a 20 seconds period, it only contains temporal information with little meta data, making it poorer, less precise and harder to use than OCU. Yet they are compatible with one another. The RT data set has 20 fields and represent around 7GB of data per year with daily file sizes varying between 5 to 35 MB.

It is worth noting that both RT and OCU data sets contain commercial and deadhead trips. A bus trip is commercial when the bus is transporting passengers within a bus line

2. Terrestrial Trunked Radio

service. Deadheads are network management trips that do not transport people, e.g trips between a deposit and the origin terminal of a bus line. This means that any kind of trip can be analyzed using these data sets.

2.3 Data Cleansing Strategy

In the OCU data set, readings for which bus speed was lower than 1 km/h or higher than the legal speed limit of 70 km/h, which respectively are low speed limit we defined and legal maximum speed for buses in France, can represent up to 5.4% of the data of OCU. Such information is absent from the RT data set that contains no metrics.

Still, considering the richer information available in OCU, it appears that it is the only production data set we have that can directly be used as a source for prediction tasks. To evaluate how much error correction is needed before prediction can be done efficiently, we ran 4 experiments, with the following data sets:

- H0) raw OCU;
- H1) Cleaned OCU, by applying business rules:
  - Illegitimate bus speed ($< 1km/h$ or $> 70km/h$) are deleted;
  - null meta-data values that cannot be inferred are deleted;
- H2) inter-stations
  - Enriching OCU by merging with RT to complete missing timestamps for stop arrival and stop; departure recordings
  - Joining with REF to update empty distances and add meta-data;
  - Identifying and isolating trips to infer primary keys (trip departure, bus line, ...);
  - Recomputing commercial speed metrics with the new raw values;
- H3) Cleaned H2 using the same rules as H1.

Building this data set is costly because RT does have little metadata. Hence meta data such as starting hour of each trip, trip identification, etc. must be inferred "trip by trip", making N computing tasks with N being the number of trips to recompute. Table 2.1 shows an overview of the process on a sample of data for a trip of the bus line 51. It shows that the base OCU data set is rich and ordered while base RT is poor and messy. Base RT needs to be ordered to extract individual trips and process them. Then the trips primary keys that include different fields such as trip departure information are rebuilt according to the minimum stop order within the trip, allowing the identification of missed

trip starts (delayed or unplanned). Finally, the resulting enriched RT is merged with base OCU, keeping data from base OCU when it is valid.

H0 and H2 data sets are available here[3] in an anonymized and reduced version, in order to make the experiment reproducible.

3. https://github.com/Tritbool/STAR_datasets

Table 2.1 – H2 Creation process overview

Base OCU (lost tuples, missing values, erroneous measures)

line	time_start	direction	stop_id	previous_stop_id	arrival	departure	travel_time	distance	speed	...
?	?	?	?	?	?	?	?	?	?	...
?	?	?	?	?	?	?	?	?	?	...
51	17:34:00	A	1000	?	17:36:32	17:36:40	43	317	26.5	...
51	17:34:00	A	2346	1000	17:37:41	17:37:51	79	380	17.3	...
51	17:34:00	A	2347	2346	17:38:55	17:39:03	82	430	18.9	...
51	17:34:00	A	2356	2347	17:42:18	17:42:18	203	521	9.2	...
51	17:34:00	A	1981	2356	lost	lost	201	0	768.0	...
...	...	...	...	...	...	...	...	...	...	...

●

Base RT

line	direction	stop_id	arrival	departure	...
51	A	1000	17:36:32	17:36:40	...
3	R	3452	11:21:22	11:21:25	...
51	A	2347	17:38:55	17:39:02	...
12	A	1234	13:21:02	13:21:17	...
51	A	4021	17:32:00	17:34:00	...
1	R	1000	8:02:00	8:02:21	...
51	A	1001	17:35:07	17:35:18	...
1	A	1287	8:03:00	8:03:01	...
51	A	2356	17:42:18	17:42:18	...
51	A	2346	17:37:41	17:37:55	...
51	A	1981	17:42:52	17:43:01	...
...	...	...	...	...	...

⇓

Enriched RT (requires to locate & synchronize trips using REF database)

line	time_start	direction	stop_id	order	arrival	departure	travel_time	distance	...
1	8:02:00	A	3657	1	8:02:00	8:02:21	0	0	...
1	8:03:00	R	1287	4	8:03:00	8:03:01	-1	457	...
3	11:00:00	R	3452	16	11:21:22	11:21:25	77	298	...
51	17:34:00	A	4021	1	17:32:00	17:34:00	0	0	...
51	17:34:00	A	1001	2	17:35:07	17:35:18	78	316	...
51	17:34:00	A	1000	3	17:36:32	17:36:40	43	317	...
51	17:34:00	A	2346	4	17:37:41	17:37:55	49	267	...
51	17:34:00	A	2347	5	17:38:55	17:39:02	69	430	...
51	17:34:00	A	2356	6	17:42:18	17:42:18	67	521	...
51	17:34:00	A	1981	7	17:42:52	17:43:01	43	282	...
...	...	...	...	...	...	...	...	...	...

⇓

Base OCU × Enriched RT

line	time_start	direction	stop_id	order	arrival	departure	travel_time	distance	speed	...
...	...	...	...	...	...	...	...	...	...	...
51	17:34:00	A	4021	1	17:32:00	17:34:00	0	0	-1.0	...
51	17:34:00	A	1001	2	17:35:07	17:35:18	78	316	14.6	...
51	17:34:00	A	1000	3	17:36:32	17:36:40	43	317	26.5	...
51	17:34:00	A	2346	4	17:37:41	17:37:51	79	380	17.3	...
51	17:34:00	A	2347	5	17:38:55	17:39:03	82	430	18.9	...
51	17:34:00	A	2356	6	17:42:18	17:42:18	203	521	9.2	...
51	17:34:00	A	1981	7	17:42:52	17:43:01	43	282	23.6	...
...	...	...	...	...	...	...	...	...	...	...

2.4 Quality Experiments

2.4.1 Experimental setting

Table 2.2 – data sets properties

data set	Population	Average speed	Speed standard deviation	Minimum speed	Maximum speed	Speed error rate
H0	16793293	20.31 km/h	11.6 km/h	-996 km/h	130 km/h	5.4%
H1	15880770	21.33 km/h	9.76 km/h	1 km/h	70 km/h	0%
H2	17496142	20.77 km/h	10.3 km/h	0 km/h	70 km/h	2.6%
H3	17038432	21.33 km/h	9.9 km/h	1.1 km/h	70 km/h	0%

data set	Number of inter-stations	Average Population per inter-station
H0	10262	1636
H1	8398	5041
H2	3363	5234
H3	3329	5149

Total H0,H1,H2 and H3 common inter-stations
3119

Table 2.2 shows the statistics and data quality variation of H0, H1, H2 and H3 data sets. It contains statistics about the bus network (i.e every inter-stations in each data set) such as the total population of the data set (number of recordings), average commercial speed, commercial speed standard deviation, minimum and maximum commercial speed metrics in the data set (outliers), commercial speed error percentage (metrics that are under 1 km/h or over 70 km/h), number of identified inter-stations in the data set and their average number of recordings. Finally, this table contains the number of inter-stations that are common to H0, H1, H2 and H3, that is the inter-stations core we use for the study.

Below is the global population variation. In accordance with **Table 2.2**, H2 and H3 are more populated than H0, and H1 and filtering a data set reduces the population of

the resulting data set in all the cases.

— H0 → H1 : - 5,434 %
— H2 → H3 : - 2,616 %
— H1 → H3 : + 7,29 %

All of the data cleansing steps were done using a 4 cores 8 threads CPU @ 3.0 GHz with 32GB of ram @ 2933 MHz laptop running a 64 bits version of Ubuntu 19.04

— Preparing a sample of H0 or H1 data for a single day takes less than 5 minutes of computing and manual manipulations.
— Preparing a sample of H2 or H3 data for a single day takes around 2 hours of computing, involving a bottleneck of 1h+ for the RT preparation before merging with OCU.

2.4.2 Commercial speed prediction experiments

As set in the introduction, our stated goal is to measure the impact of data quality level on the prediction models's precision. For H0, H1, H2 and H3 we learn and predict bus commercial speed over 3119 common inter-stations given built-in features:

— Bus line ID
— Type of day
— Period in the day
— Month in the year
— Holidays time

All the features were selected knowing that the most important one we have built-in is the period in day[39], then normalized and missing values were imputed using K-Nearest Neighbours imputation, with K set to 10. Thus, the missing data is imputed using the data of the 10 nearest neighbours of the erroneous row, according to which class of neighbours is more numerous within the bench of neighbours.

We learned the speed on the different inter-stations of the network in such a manner that the output is the predicted bus speed for a given bus line at a given period of the day on a given and already known inter-station. Hence, we predict the speed using only temporal and categorical built-in features, ignoring the spatial information [4] knowing that inter-stations are only 607 meters long in average, begin and end with a bus stop.

Using the data sets presented before, we trained a set of 6 different prediction models

4. That we do not have anyway.

from the Scala SMILE framework [5] in its 1.5.2 version:

1. Random Forest;
2. Decision Tree;
3. Lasso;
4. Bayesian Ridge;
5. Gradient Boosting;
6. Ordinary Least Square.

We chose these regression models because they are common and long known as well as for their ease of use and overall performance. We used 10-fold cross validation on each model to configure the models hyper-parameters and kept the best resulting prediction's RMSE amongst them.

The Root Mean Squared Error is the standard deviation of the predictions errors :

$$\sqrt{\frac{1}{N}\sum_{1}^{N}(Y_t - \hat{Y}_t)^2}. \tag{2.1}$$

N is the population of the test data set, Y_t the observed commercial speed and $\hat{Y}_t$ the predicted commercial speed. A low RMSE (i.e near 0) indicates a good prediction accuracy. The RMSEs results are compared in Figures 2.1, 2.2, 2.3, 2.4 and 2.5.

Figures 2.1 to 2.4 compare the resulting RMSE for different pairs of data sets, respectively H0-H1, H2-H3, H0-H2 and H1-H3. On x axis are presented the inter-stations ordered by name (for a better readability of the legend, only a few names are displayed over the 3119 inter-stations). On y axis are presented the RMSE values for each inter-station.

Figure 2.5 summarizes the RMSEs of H0, H1, H2 and H3 in order to facilitate the analysis of results

Also we compared RMSEs normalized on scatter indicator of input data (here standard deviation, and mean) in Tables 2.4 and 2.5. The nearer to 0 it is, the better the fit is. De facto, it is harder to get a normalized RMSE close to 0 normalizing with standard deviation than with mean.

The prediction models were trained on a different and more powerful machine with 64 cores @ 3.4Ghz and 64 Gb RAM @ 2933Mhz, running Manjaro 18.1 in its KDE version.

Computation time for the 3119 inter-stations of each data set is around 14 hours and detailed as follows:

5. http://haifengl.github.io/

— 3h30 data preparation;

— 10h30 of training for 3119 inter-stations * 6 models.

During prediction experiments, some inter-stations training would fail, either because of data issues (like insufficient amount of data, malformed data), or tweaking issues (as we tweaked each model only once in accordance to the average statistics of the inter-stations). **Table 2.3** shows the failing statistics for each input data set.

Table 2.3 – inter-stations learning failures

data set	Amount of failures
H0	160
H1	160
H2	126
H3	186

Figure 2.5 shows the variation of prediction accuracy between the different data sets.

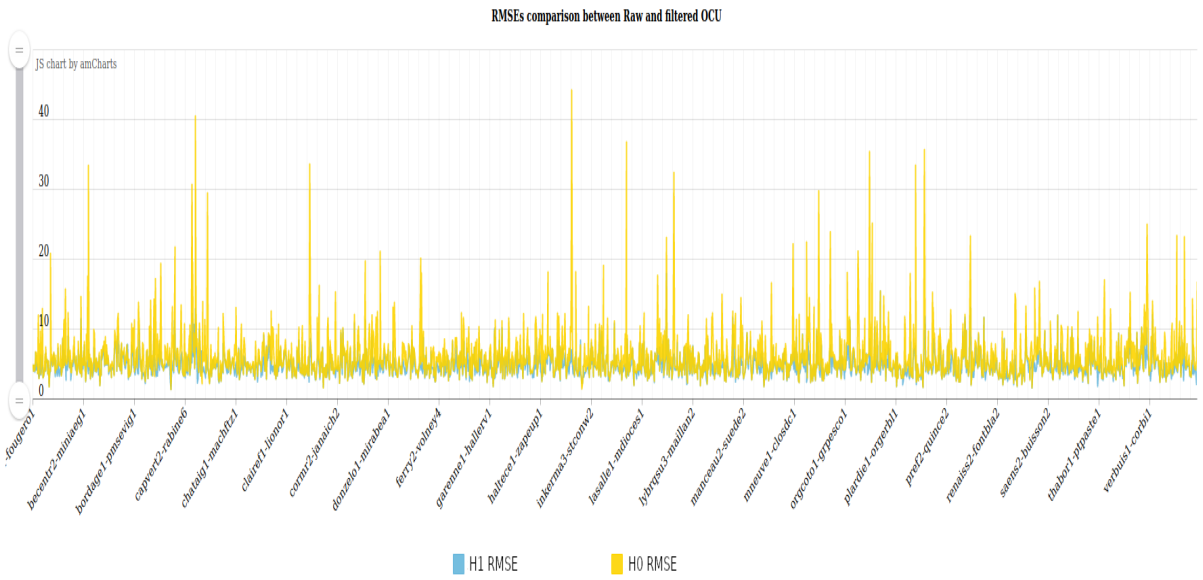

Figure 2.1 – RMSE variation over 3119 inter-stations common to H0 and H1

2.5 Discussion

Figure 2.1 shows the RMSESs of the inter-stations of H0 and H1. The content of **Figure 2.5** supports the fact that the quality enhancement done from H0 to obtain H1

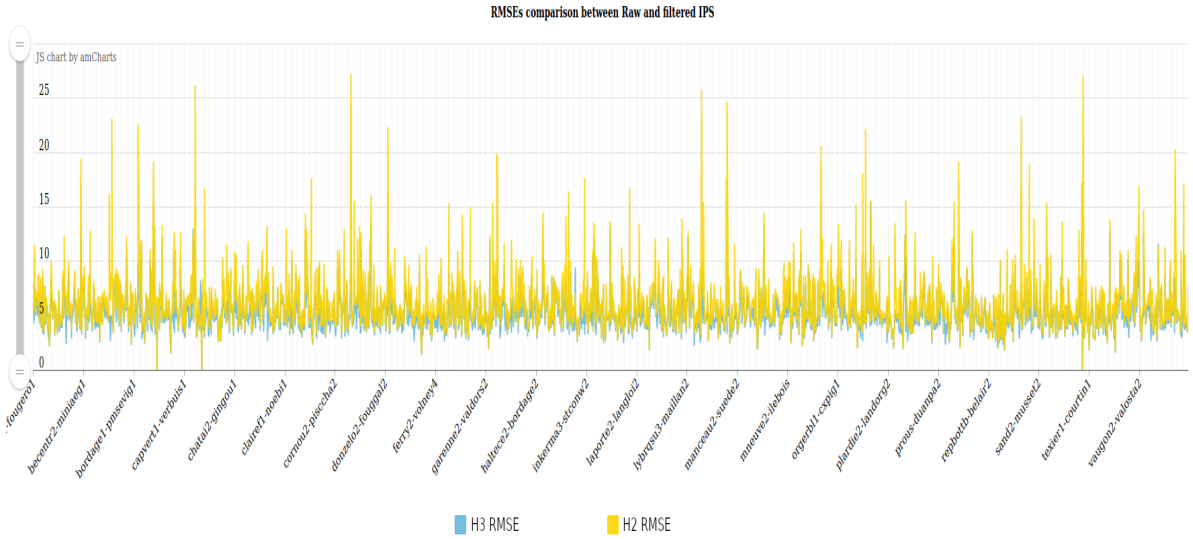

Figure 2.2 – RMSE variation over 3119 inter-stations common to H2 and H3

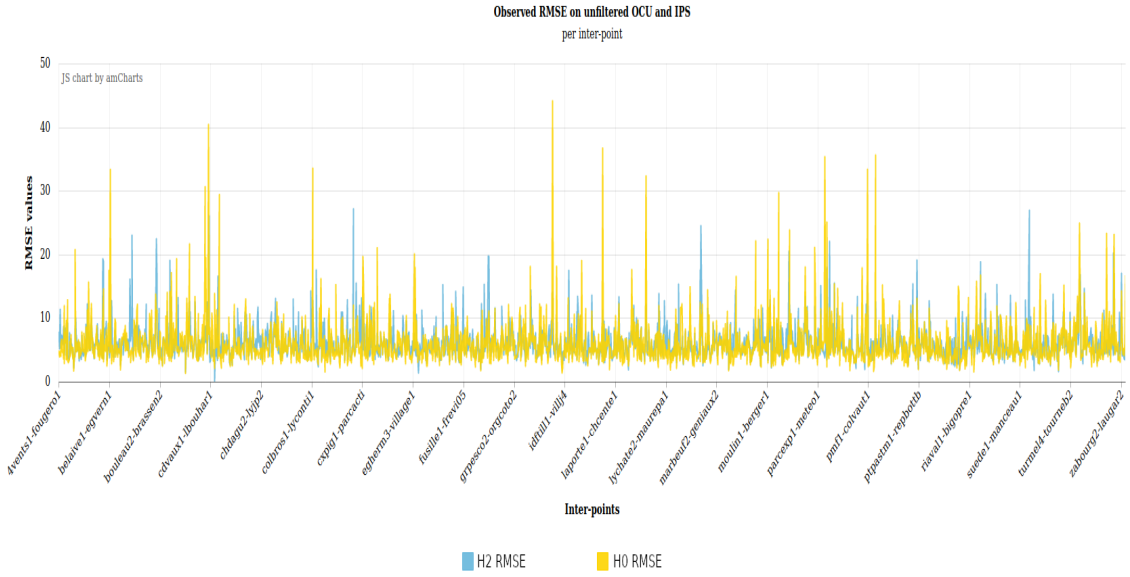

Figure 2.3 – RMSE variation over 3119 inter-stations common to H0 and H2

implies a statistically significant boost in predictions with the average RMSE of predicted H1 being 21.34% smaller than predicted H0. In the same way, the 95th percentile in predicted H1 is 33.09% smaller than the 95th percentile of H0. In the same way, the standard deviation of the RMSE of H1 benefits from a 57.87% drop in comparison to the

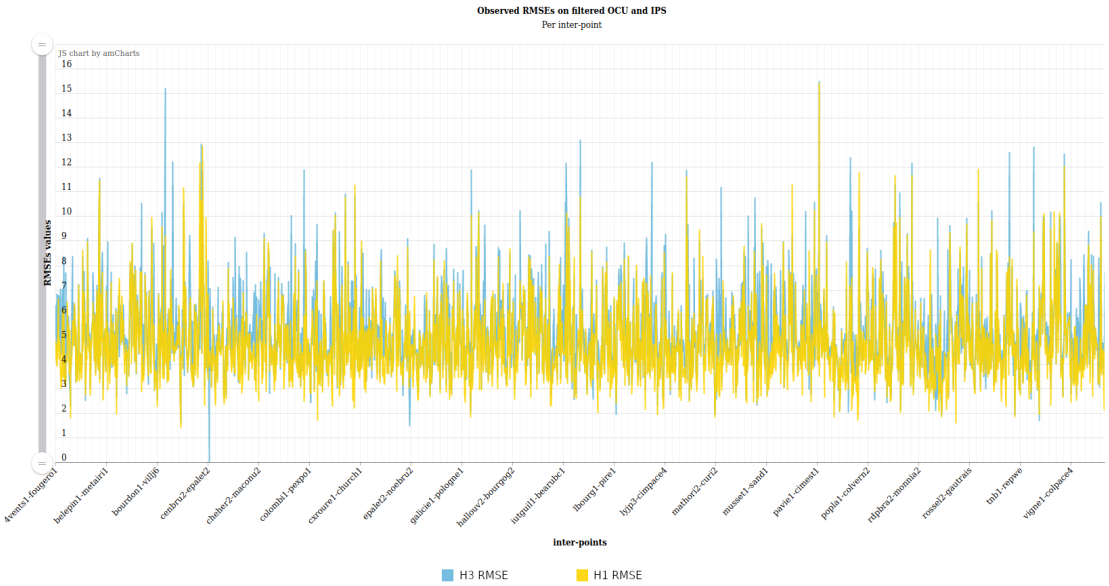

Figure 2.4 – RMSE variation over 3119 inter-stations common to H1 and H3

Table 2.4 – NRMSE of H1

SD normalized RMSE	Mean normalized RMSE
0.84	0.2

standard deviation of the RMSE of H0..

In **Table** 2.2, the filtering of H0, yielding H1, shows that the average population per inter-station dramatically raises, while the number of represented inter-stations drops by 18.16%. We note that the global population varies in the exact same proportion as the input data error rate. This is due to the fact that the amount of faulty data in the global population is equal to the error rate of the global population.

Figure 2.3 shows the RMSESs of the inter-stations of H0 and H2. **Figure 2.5** contains insights that let us think that the quality enhancement done from H0 to obtain H2 does not really enhance the prediction precision. Indeed, with an increase of 1.14% of the average RMSE from H0 to H2, it seems to be of little interest when contrasted with the

Table 2.5 – NRMSE of H3

SD normalized RMSE	Mean normalized RMSE
0.91	0.23

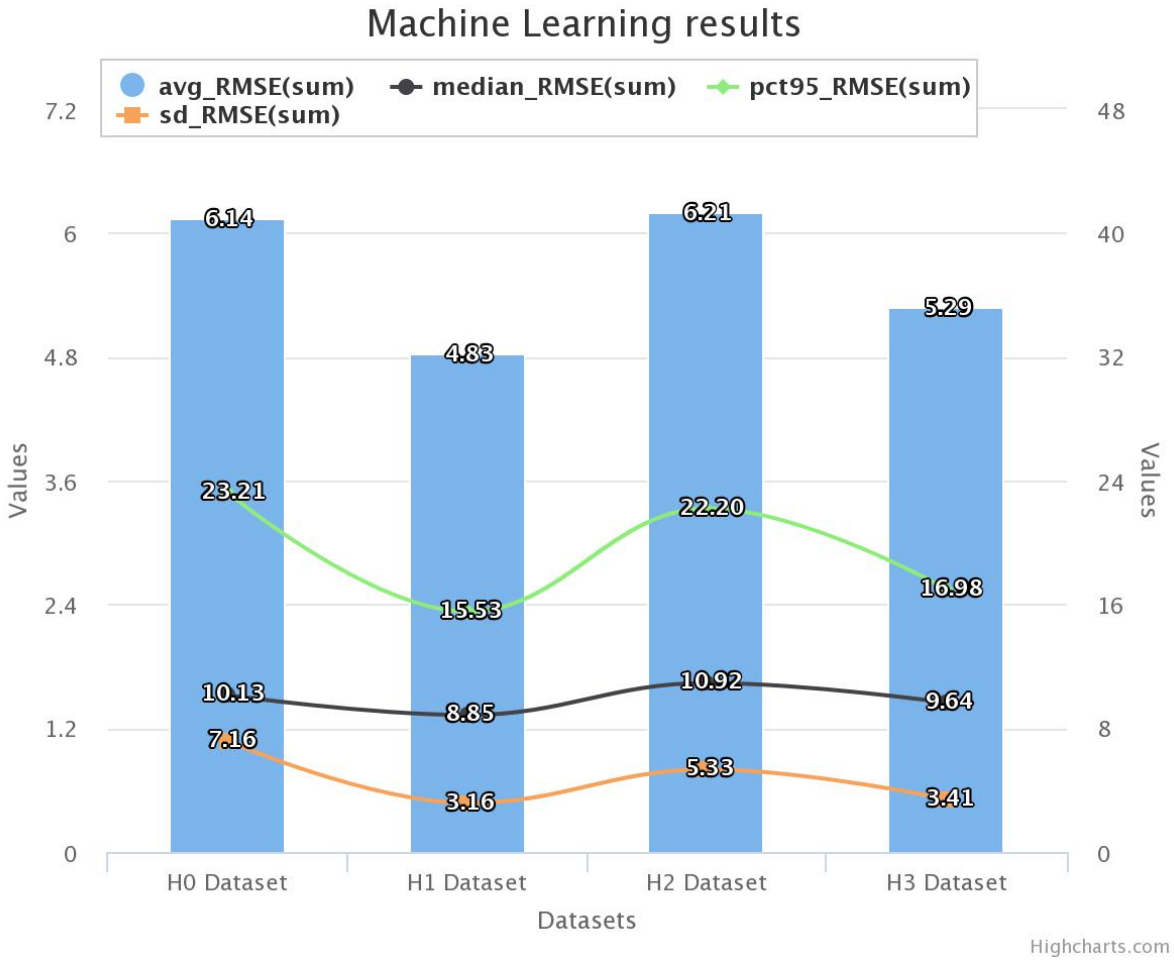

Figure 2.5 – Prediction results comparison chart

computational time needed to process a single day of data for H2. Indeed, processing a day of data for H2 is 24 times longer than computing a day of data for H0.

However, as shown in **Table** 2.2, the speed error rate is lowered by around 51,85% from H0 to H2. In the same way, H0 outliers are far more distant from the mean than H2's. On the other hand, it is worth noting that even if the average inter-station population is more than doubled from H0 to H2, the standard deviation of the commercial speed only varies by around 10%. Also, the mean speed is quite steady between those two data sets. We observe that the average population per inter-station of H1 and H2 is not much different with only 2% of variation. Added to the heavy computing needed to build H2 from H0, as stated before, those results make it quite obvious that the cleansing from H0 to H2 is somehow an overkill compared to the cleansing from H0 to H1 in terms of data

set statistics and prediction performance.

Figure 2.5 confirms that H2 is somehow not suitable for commercial speed prediction, as stated before. Also, **Figure** 2.5 highlights the evidence that the cleansing from H2 to H3 is worthy, filtering H2 being less costly in computing time than building it. Filtering a day of data from H2 yielding H3 takes a few minutes. The global population varies in the exact same proportion as the error rate, following the same rules and trends of the population evolution between H0 and H1.

Finally, **Figure** 2.5 shows that even if H3 is more populated than H1 (cf. Table 2.2), the enhancement of the standard deviation and 95th percentile of RMSEs are not sufficient to make it worth the cost to compute H2 and H3 for prediction. Worse, the average RMSE's of H3 increases even if the average speed and standard deviation of H1 and H3 are similar as stated in Table 2.2, and the normalized RMSEs in **Tables** 2.4 and 2.5 supports the idea that H1 is the best choice for prediction in this specific context.

2.5.1 Threats to validity

Construct

In this particular context, construct validity implies that the data chosen for the experiment is representative enough regarding the original data and the data quality rules created on purpose. Also the prediction models should behave evenly when an experiment is led several times using the same datasets.

Prediction models were not tuned using grid search because of time issues and compatibility in the SMILE API. This implies that the results could be better with enhanced hyper-parameters, yielding less training failures. In our investigation, we found that many of the inter-stations for which prediction failed were poor in data. Also, the amount of failures in our study represent less than 6.5% of the 3119 inter-stations we had in average. Recovering all of the failures could change the overall result, yet, probably not dramatically even with a better tweaking of hyper parameters.

We assumed that the inter-stations were small enough not to consider the spatial information. May be it would be interesting to confirm whether this was right or not in our further work. Usually [23], the ratio of elements (such as traffic lights, stops, etc) per kilometer is considered to estimate their impact on bus commercial speed. Our inter-stations being smaller than 1 km, they probably are short enough for those elements not to impact the commercial speed significantly. However, adding the spatial data to our

data set in further studies is needed to confirm this claim.

inter-stations that are exclusive to each data set were not tested and may have changed the interest of H2 and H3, hence the global result of the study. If one wants to learn and predict commercial speed over the whole network (commercial and deadhead trips), then H0 would be the way to go according to the fact that it contains far more inter-stations information. However, if one wants to work on commercial trips only, H1 or H3 would be better. Hence, depending on the goal, the outcome might be different. In our case, we aimed at predicting the commercial speed regardless of the commercial or deadhead aspect of the input trips data.

Internal threats to validity

We selected features among dozens of fields based on what others studies have used to predict commercial speed. It is unlikely that other features that the one we used would have significantly enhanced the prediction results, yet it is still a threat to consider.

We assumed that the 20 seconds temporal resolution of the data in RT has no specific impact on the bus travel time, the amount of trips smoothing it out. We somehow validate that there is indeed a smoothing effect when we compare the average speed of H1 and H3 in Table 2.2 which are equal. However the standard deviation evolution shows that the 20 second resolution has an impact, even if it is quite low. If we had to recover more than 13% of data from RT, may be the resolution would have a greater impact unless the whole data set was recomputed from RT only.

We did not control the data quality of timestamped data and odometer distances metrics. More generally, we have no mean to assess the accuracy level of the AVL that provides us with this data. Even if we cannot control the data upstream, we observe that the data consistency is quite good: as already discussed, H0 only has 5.4% of errors (cf Table 2.2). If the odometer were to fail massively, we would probably observe a rise in speed error rate, making it clear that we should not use this data as an input for our experiments.

External threats to validity

Cleansing rules were chosen based on what is known about legal bus speeds in France and it is possible that the filtering could be better done. Typically, each inter-station would probably be predicted more accurately if they were to have their data filtered against their own specific legal maximum speed, which was not possible to consider in this study. Based

on the fact that bus driver are urged to respect speed limits in the different areas the bus travel through and that this is a professional condition, it is probably legitimate to assume that the speed recorded within the different inter-stations stays within the legal limits in almost all of the cases. Yet a significant rise of the commercial speed would be an indicator of driving issues.

This study was based on the bus data of a single city, Rennes (France), for a single period of time. We mitigated the single period of time issue by running the same experiment in varying periods, without noticeable impact on our results. The Rennes metropolis is typical of Western European cities, with a crowded medieval downtown, less crowded suburbs, and some reserved ways for public transportation, so we foresee that most of what we have learnt here could be applied to similar cities. The same goes for the data quality issues: as we have seen before, most of Public Transportation Information Systems suffer from similar data quality issues. Still before being able to generalize our results, it would be necessary to run similar experiments with other cities bus networks. This could be easy to do since our data processing methods could be reused out of the box for a new data set.

2.6 Conclusion

In this chapter we worked on 4 different data sets that gather 6 months of data from the Public Transportation Information System of the bus network of Rennes, France. We used different data quality enhancing techniques and compared the resulting statistics, and prediction usability of those data sets. It appeared that, in this context, data quality is needed to ensure acceptable data sets statistics and prediction accuracy. However, over qualifying the data implies a high ratio of computational cost against performance gain, making it unadvised for non-production purposes like prediction experiments. Finally, one would have to define a threshold for the ratio between computation cost of data quality enhancement versus gain in prediction accuracy over which it is considered counter productive to enhance more the data quality of the input data set.

In the following chapter, we consider exogenous data sets, that contain meaningful data for commercial speed interpretation.

Chapter 3

Data analysis and assessment of bus commercial speed impacting factors at inter-station level

3.1 Introduction

Commercial speed, that is the speed of the bus felt by the passengers while they travel, is among the major indicators used by operators to evaluate the efficiency of bus lines and assess the state of health and quality / attractiveness of a bus network. This measure is computed using the total travel time over any section on which the indicator has to be computed, including boarding and alighting time at bus stops, traffic perturbation, and any time related factors that impact the travel time, hence the speed of buses as stated by [23, 55, 32, 44, 14, 15]. Studies have been made on what to act on to both understand and control bus commercial speed fluctuations. It appears that time in day is a major factor (rush hour, etc.) but the list of factors is probably not exhaustive, especially since the commercial speed is measured from buses operating in a rich and complex environment [42, 66, 67, 65, 35, 55]. However, different previous works have enligthened some other impacting factors. Hofmann et.al. [32] Observed the weather impact on buses performance, especially regarding rainy conditions over the whole bus network of an anonymous Irish city. They reported that the buses travel time seems to be globally higher under heavy rain conditions. Shared bicyles services impact on public transport travel times has been explored by Jäppinen, Toivonon and Salonen [35]. They assessed this impact in the city of Helsinki, Finland. Traffic influence on buses travel time have also be reported to be prominent, as mentioned by Mazloumi [44] and Cortes [14]. However, the gathering of valuable traffic data seems to be a possible challenge. Tirachini et al. [65] did a survey on how the fare payment system, bus floor level and passengers age can impact buses dwell time, thereby bus commercial speed. Their work reported some

correlations among those factors and buses dwell time. Finally Fernandez [23] and Valencia [69] built some analytical models that leverage bus network infrastructure variables such as traffic lights, bus stop spacing in and out of bus corridors, vehicles hardware, etc., thanks to which they could build accurate (when properly tweaked) predictive analytical models. This previous work has yet to be compiled in a single macro survey, that could summarize the importance of different identified bus commercial speed impacting factors, at a bus network scale. In this chapter, we propose to evaluate a set of known factors, for which we gathered data either at the bus network scale, or a distributed sample over the bus network, from various and heterogeneous data sources. In particular, we observe data of the lockdown that occurred in France between the 17th of march 2020 and may 12th 2020 during which the bus network was in degraded mode and the traffic, ridership and global mobility were drastically diminished, and if these changes are visible on the bus commercial speed. We had access to the internal data of the bus network of the city of Rennes, a medium sized European city (∼210'000 inhabitants in the main city, ∼450'000 inhabitants at metropolis scale as of 2018). This network, named STAR [1], is a star-shaped urban transportation network based on a central subway line, and a wide bus network that serves both the city of Rennes and all its suburban areas. In total, the bus network covers more than $550km^2$. The transportation network is managed by the Keolis Rennes company that is responsible for the main subway, a transportation service dedicated to disabled people, bikes (rent and sharing), and a total of 116 bus lines over which more than 600 buses can be traveling during rush hours.

The remaining of this chapter is organized as follows: i) We expose our datasets, their properties, qualities and default. ii) We analyze the impact of the factors we gathered data against the bus commercial speed using correlation tests and visual analytics, including a closer look at the data of the first COVID19 lockdown. iii) We discuss our results and findings, and we conclude in iv).

3.2 Data sources

3.2.1 Automatic Vehicle Location data

The bus network information system is made of several independent subsystems including (among others) fuel manager, smart card data and an Automatic Vehicle Location

1. https://www.star.fr/

(AVL) systems. The latter yields large amount of stop to stop fine-grained data (inter-station) both in real and delayed time. This data contains metadata to identify bus line, direction, vehicles, etc. and timestamped and located information such as buses speed, travel time, dwell time, start time, etc. A year of data typically weighs around 20GB, cf. chapter 2.

3.2.2 Ridership

We gathered ridership data using the smart card data system of the bus network. This system gathers timestamped boarding only information at each bus stop. The boarding information discriminates smart card boardings from ticket boardings (the onboard hardware centralizes tickets and smart cards validation). Hence it is possible to make different measures of the impact of each of those fare payment systems. In Rennes, tickets can be bought either in affiliated shops or directly onboard, by asking the driver who then sells them, implying a trade of cash. On the other hand, the only interaction with smartcards in buses is NFC validation when boarding, the payment being made in agencies or vending machines. The data we collected has around 20'240'000 readings and covers a seventeen months period from 2019-01-01 to 2020-05-12, over 2293 commercial inter-stations (i.e deadheads are not represented in this data).

3.2.3 OpenStreetMaps data

Bus networks are particularly intricated within the cities infrastructure they are part of. Unless dedicated equipment such as traffic signal priority or bus lanes are built, most of the time the buses are running within the traffic like any other vehicle. Hence things such as travel time (speed), delay, or even fuel consumption are dependent of the road infrastructure [20, 34]. A problem with road infrastructure is that it is quite complicated to get a satisfying and global view of it because of its richness and complexity. As an example, one could try to gather information about traffic lights all over a given city, but there is little chance that this will be achievable in a reasonable time (hundreds to thousands of traffic lights to identify and map to infrastructure used by buses). A solution to this problem is to mine online databases that are reliable, up to date, and open sourced OpenStreetMaps (OSM), is the most known database that contains road infrastructure, and that lets anyone download free dumps of the database or query its knowledge base, which is groingly qualitative [6, 24]. Moreover, its representation system is

quite complete since it allows, among other things, to represent public transport networks by linking them directly to the existing road representations [2]. Thus, we developed a tool that thereby extracts road information directly from the OSM dumps, using this built-in linking system. We could gather the following data for 1400 inter-stations :

— Number of traffic lights, stops, giveways, roundabouts, crossings;
— Proportion of road for which buses share it with bikes, are in a bus lane, are in one-way road;
— The legal speed all along the ways;
— the length of the ways.

This software however has limitations:

— The path finding is returning the smallest path. In some cases it might return an incomplete path (when the bus takes a one way loop for example);
— There might be data redundancy as long as two different trips that travels through the same roads will yield similar/identical data;
— When a non-terminal bus stop is located somewhere along a road, we arbitrarily split the road's length in two for the current and its neighbouring inter-station. However, we do keep all the traffic signals, crossing, etc bound to this road in both of the inter points [3].

3.2.4 Traffic

Traffic is told to be one of the most important disturbance generator for bus networks [23, 15]. Thus, gathering traffic data that is compatible with bus network data is mandatory to actually measure the impact of the traffic on bus speed. In the city of Rennes, there are plenty of sensors managed by the metropolis. However, the data yielded by this sensors fleet is not freely accessible. Fortunately, there is an open-data platform hosted by public services, that proposes a traffic Floating Cellular Data traffic status database with a 3 minutes time granularity [4] The data available on this service represents the status of the traffic every 3 minutes, with the following information in each row:

— **predefined location reference**: the identifier of the road section.
— **date time**: the timestamp at which the measures were taken.

2. https://wiki.openstreetmap.org/wiki/Public_transport
3. To reduce this unwanted noise, an evolution would be to split the roads according to the actual bus stop's position and try to keep the infrastructure elements that are matching the split. Please see https://gitlab.inria.fr/glyan/osm_bus_extractor for more details
4. https://data.rennesmetropole.fr/explore/dataset/etat-du-trafic-en-temps-reel/information/

— **travel time** the estimated travel time to cross the road section.
— **travel time reliability**: the reliability of the travel time estimation from 0.0 to 1.0.
— **average vehicle speed**: The average vehicles speed on the road section
— **traffic status**: the status of the traffic that can be free, heavy, congested or impossible.

One major concern with this dataset is that the sections represented in the data have to be manually identified and bound to the AVL data. Finally, we could match data for a total of 140 inter-stations, totaling 1'362'000 readings over the period 2019-01-17 to 2020-04-22.

3.2.5 Bicycles

Bicycles and other means of soft mobility such as electric scooters will be more and more present in our cities due to the ecological transition that humanity is trying to make. Furthermore, bicycles sharing systems have been shown to have an impact on buses performance [35].

The proportion of road shared with bicycles that we extracted from OSM is an interesting source to demonstrate the impact of bikes on bus speed at a large scale. To assess that, we deployed a traffic counter and proceeded to a series of measures of the number of bikes per 5 minutes on a portion of shared lanes for a 2 months period, yielding 9100 readings (including outliers).

3.2.6 Weather

We gathered weather data for a 14 months period. This data comes from a single weather station located at the St-Jacques de la Lande airport, Rennes. Its data granularity is one hour based and is mainly useful to measure the amount of rain during the last hour. It is worth mentioning that the region in which Rennes is situated is hardly affected by snowfall. The local weather can be rainy, with low amount of rain often falling. In other words significant rainfall, flooding or storms can occur at different times of the year, but they are rare events.

3.2.7 Bus hardware

The bus fleet is made out of different type of buses with different sizes: from standard to articulated, powered by electricity or fuel. Keolis Rennes has the direct responsibility for a bus fleet of 282 different buses from different manufacturers. This data comes from specific reference files, and is accurate enough to enlighten how bus hardware can yield bus commercial speed discrimination.

3.2.8 Lockdown period

We could gather all the previous data during the lockdown period that happened in France between 2020 march 17th and 2020 may 12th. This actually was a chance for us because the lockdown resulted in a massive drop in traffic, ridership and urban mobility due to the lockdown policy.

3.3 Factors impact analysis

The data we gathered have been collected, enriched and analyzed using Apache SPARK 3.1.1 and R. Those were also used to build Spearman correlation matrix and/or tables and visual analytics if relevant for particular cases. Note that the Spearman correlation coefficient p-values have all been verified to be under 0.05. In addition, it is worth mentioning that because bus networks are complex models, involving multiple relations and entities, the Spearman correlation coefficients of speed impacting factors are expected to be quite low (under +/- 0.4).

3.3.1 Weather

We made some correlation tests between our weather data and our AVL data, over a total of 8'535'000 readings. The results we obtained enlightened an absence of correlation between the amount of rain and the bus commercial speed. The fact that the average amount of non-null rainfall is around 2mm per hour [5] can explain why there is no relation between those data. On the other hand, when we focus on the 2122 readings of heavy rainfall (>= 8.0 mm per hour), we observe a very weak -0.1 negative correlation coefficient between heavy rainfall and bus commercial speed. We have to note that the few amount

5. (considered as low by Meteo France: http://pluiesextremes.meteo.fr/france-metropole/Intensite-de-precipitations.html)

of readings let us think that a survey involving more data is needed to assess that heavy rainfall has an actual impact on bus commercial speed in Rennes, as weak as it is.

3.3.2 Infrastructure

Fig. 3.1 shows the correlation matrix of the road infrastructure and bus commercial speed. The data we used to build this matrix is composed of AVL data and OSM data. It spans over fourteen months from january 2019 to march 2020, for a total of around 29'800'000 readings all over the bus network, including urban, sub urban and metropolitan areas. Table 3.1 details the matrix rows and columns names, and the input characteristics. If we take a look at the row dedicated to bus speed, name *speed*, we can see that there

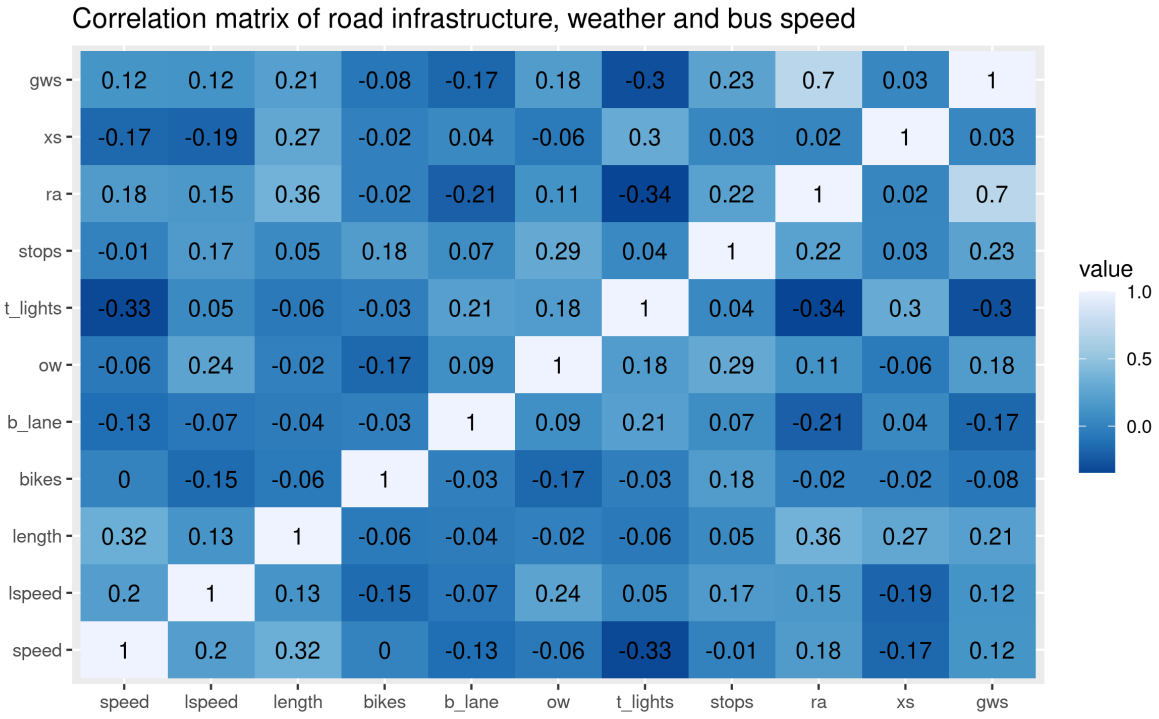

Figure 3.1 – Correlation matrix of road infrastructure, weather and bus commercial speed

are **weak negative correlations** between bus commercial speed and the growth of (in order of importance):

1. The number of traffic lights (-0.33).
2. The proportion of bus lane (-0.13).
3. The amount of pedestrian crossings (-0.1).

Column/row	**Data represented**
speed	Bus commercial speed (in km/h)
lspeed	Maximum legal speed over the inter-station (in km/h)
length	Length (in meter) of the inter-station
bikes	Proportion of road shared with bicycles over the inter-station ($\in [0.0; 1.0]$)
b_lane	Proportion of bus lane over the inter-station ($\in [0.0; 1.0]$)
ow	Proportion of one way road over the inter-station ($\in [0.0; 1.0]$)
t_lights	Number of traffic lights over the inter-station
stops	Number of stops signals over the inter-station
ra	Number of roundabouts over the inter-station
xs	Number of pedestrian crossings over the inter-station
gws	Number of giveways signals over the inter-station

Table 3.1 – Fig. 3.1 related data and characteristics

According to this, traffic lights are, by far, the the first trigger of a lower bus speed, followed by the proportion of bus lane and number of pedestrian crossings. Some very low negative correlation coefficient are also visible for:

1. The proportion of one way roads (-0.06).
2. The number of stops signs (-0.01).

It appears that the impacts of the proportion of one way roads and the number of stop signs on the bus commercial speed are negligible. However we observe one unforeseen and counter-intuitive result that is the negative correlation between the bus commercial speed and the proportion of inter-station that benefits of a bus lane. We should expect dedicated bus lane to enhance bus speed as long as it helps to get rid of the influence of many disturbances like, e.g. traffic, pedestrians or complex crossroads. Thus a positive correlation is expected here. One way to get a clue of what yields this is to take a closer look at the row dedicated to bus lane and traffic lights in Fig. 3.1. One should see that there is a weak positive correlation between the proportion of bus lane and the amount of traffic lights (+0.21). That is to say that, in Rennes, the creation of bus lane seems to come along with the building of traffic lights. Considering that traffic lights are the major reasons of lower bus speed in our data, this thereby can give a hint of the reason why bus lanes seem to yield a lower bus speed. Moreover, there is a very weak negative correlation (-0.07) between the proportion of bus lane and a lower maximum legal speed. This can be explained by the fact that bus corridors are often built in downtown areas in which very low speed policy (30 km/h) is applied. Finally pedestrian crossings seem to

be very lowly correlated to the lowering of bus commercial speed (-0.17), hence the use of pedestrian crossings should also be reduced on dedicated bus lanes if this came to be a valid hypothesis.

In the contrary, we observe **weak positive correlations** between bus commercial speed and the growth of (in order of importance):

1. The inter-stations length (+0.32).
2. The inter-stations maximum legal speed (+0.2).
3. The number of roundabouts (+0.18).
4. The number of giveway signs (+0.12).

The positive correlation between inter-stations lengths (also known as bus stops spacing) confirms what the literature already tells [23]. In the same manner, the rise of the maximum legal speed seems to positively impact the commercial speed of buses. The existence of what seems to be a very weak correlation between roundabouts (that often come along with giveway signs) enlightens the hypothesis that roundabouts should be favored instead of traffic lights when creating a crossroad that involves the passing of buses.

Bus hardware

We used 22'512'158 readings from our AVL data enriched with the buses hardware metadata, covering january 2019 to late may 2020. We were able to classify data using buses manufacturer/power source, and size (13 meters long standard vs 18 meters, articulated). Table 3.2 summarizes the speed variation and data distribution between each manufacturer, bus size and power source.

Bus brand/power/length	**avg bus commercial speed**	**% of data**
A (fuel)	21.0 km/h	9.5%
B (fuel)	19.0 km/h	75.5%
C (fuel)	19.2 km/h	12.3%
D (electric)	16.8 km/h	2.7%
13 meters	20.6 km/h	37.8%
18 meters	18.3 km/h	62.2%
Fuel	19.2 km/h	97.3%
Electricity	16.8 km/h	2.7%

Table 3.2 – Bus hardware data summary

We can see that bus built by manufacturers B and C represent the majority of the buses that are used on the network (87.8% of readings) and have an average commercial speed of around 19 km/h. on the other hand, buses from manufacturer A yield the highest commercial speed in the sample (21.0 km/h). Electrical buses built by manufacturer D yield the lowest commercial speed, reaching only 16.8 km/h in average. If we take a look at the length of the buses, we can see that most of the readings are generated by 18 meters long, articulated buses (62.2% of readings) while smaller buses are less used in the network (37.8% of readings). The articulated buses apparently have a lower commercial speed (18.3 km/h) than standard buses (20.6 km/h). Finally, we can observe a gap of commercial speed between fuel powered buses (19.2 km/h) and electrical buses (16.8 km/h).

Our observations show that if we create groups of readings from the data using buses hardware, variations in bus commercial speed appear. However, this is yet erroneous to say that the buses hardware is actually responsible of the variations without further observations including:

1. The distribution of buses on the network, for buses that are reserved for downtown areas vs buses that are reserved for express / suburbans areas will thereby yield a constant bias in the data.
2. Specificities of the driving of some buses like, e.g. electrical buses, that benefit from regenerative braking [31], inciting the drivers to be more proactive.

Bicycles

Bicycles tend to be more used during spring and summer days, when the weather is warmer and less rainy. Thus, we targeted a portion of data for the period that goes from June 2019 to October 2019 from our dataset built upon AVL and OSM data, on urban areas only. This represents a total of 4'526'000 readings. We observed a very weak correlation (-0.06) between the portion of inter-stations that are common to bicycles and buses and the bus commercial speed. This correlation is computed using all the data available, including normal roads, bus lanes and bicycles lanes. Hence a better indicator would be to know if bicycles should be avoided on dedicated bus lanes or not. We led a survey as described in section 3.2.5 for which the results are visible in Fig. 3.2 It is a mixed boxplot-line chart, with boxes width varying with the size of the data sample they represent. The correlation coefficient is of -0.26 and is way more important in this survey than over the whole network. This suggests that there is a chance that the presence of

bicycles in bus lane is deleterious for the bus commercial speed. Once again, surveys on other areas have to be done in order to make this claim valid on other areas than the place the survey was led on.

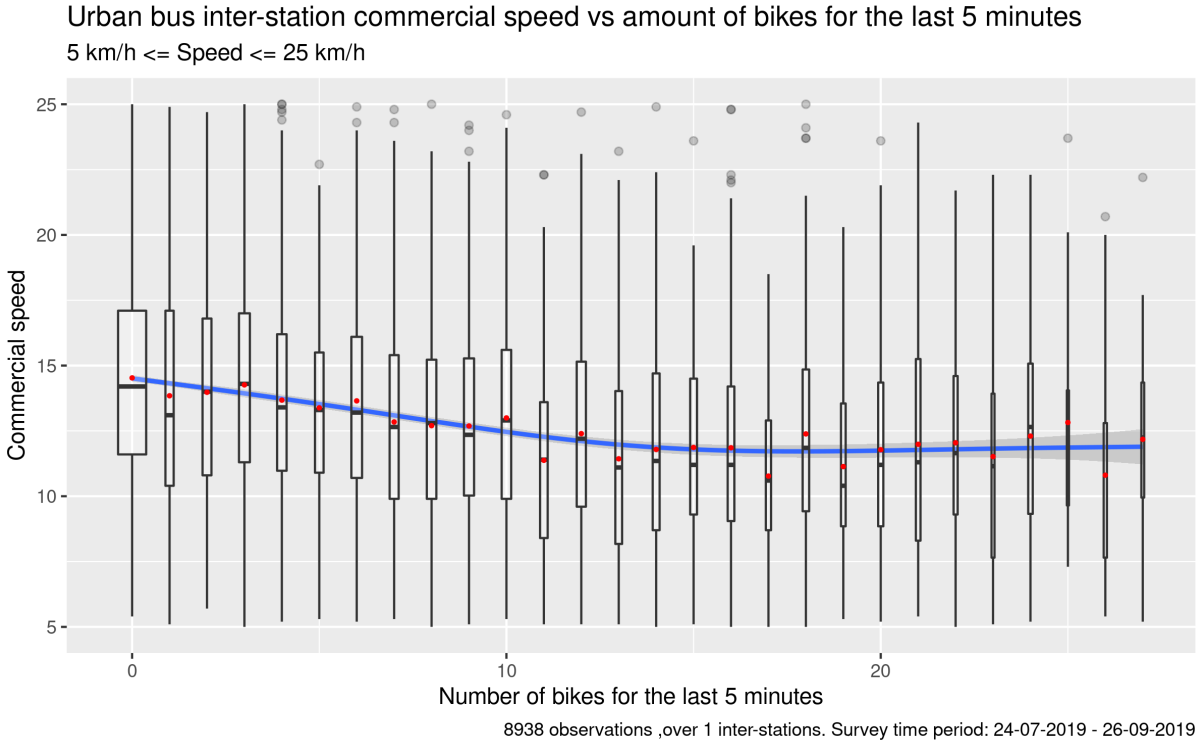

Figure 3.2 – Chart of bikes per 5 minutes vs bus commercial speed

3.3.3 Traffic

Nominal

We call nominal period the time period out of lockdown, when the bus network was in its nominal state, with all the day to day activities of the city. Using the 1'246'000 readings made over 140 inter-stations we collected for the nominal period between may 2019 to march 2020, we could build the following results. Fig 3.3 shows the correlation matrix of bus commercial speed and the traffic load and average speed of vehicles. As we can see, there is a weak negative correlation (-0.25) between the bus commercial speed and the traffic load. On the other hand, there is a weak positive correlation (+0.29) between the bus commercial speed and the average speed of the vehicles. This totally makes sense as long as when the buses are inserted in the traffic flow, their speed varies with the average

traffic speed, thereby, when the traffic gets heavy, the vehicles speed gets lower along with the bus commercial speed, and vice-versa. Table 3.3 represents summarizes the points of interest in the data we collected for the nominal period. As we can see, the average bus commercial speed in data is of 20.4 km/h and the average traffic load is quite important with a 33% average load. Moreover, the amount of data for which readings shows that traffic is starting to get heavy (>= 50%) represent more than 20% of the data we get. Finally 2.6% of the data are corresponding to jam readings.

avg bus commercial speed	**avg traffic load**	**avg speed on road**	**traffic load > 50%**	**traffic load = 100%**
20.4 km/h	33%	37 km/h	21.5% of data	2.6% of data

Table 3.3 – Traffic data summary for nominal period.

Figure 3.3 – Correlation matrix of bus commercial speed and traffic for a nominal time period.

Lockdown

Using the 116'532 readings made over 135 inter-stations we collected for the nominal period between April 22nd 2019 to may 10th 2020, we could build the following results. We can see on Fig. 3.4 that the correlation we observed for the nominal period are still present, yet the coefficients are much lower. In the same way, if we take a look at Table 3.4 we can observe that not only the average bus commercial speed is more than 3km/h higher than in nominal condition, but also that the average traffic load as dropped by 39.4% while the average speed as increased by 5 km/h. In addition, we observe that the amount of readings for which the traffic load is at least 50% as been divided by more than 2 and that traffic jams have almost disappeared with a 73% drop.

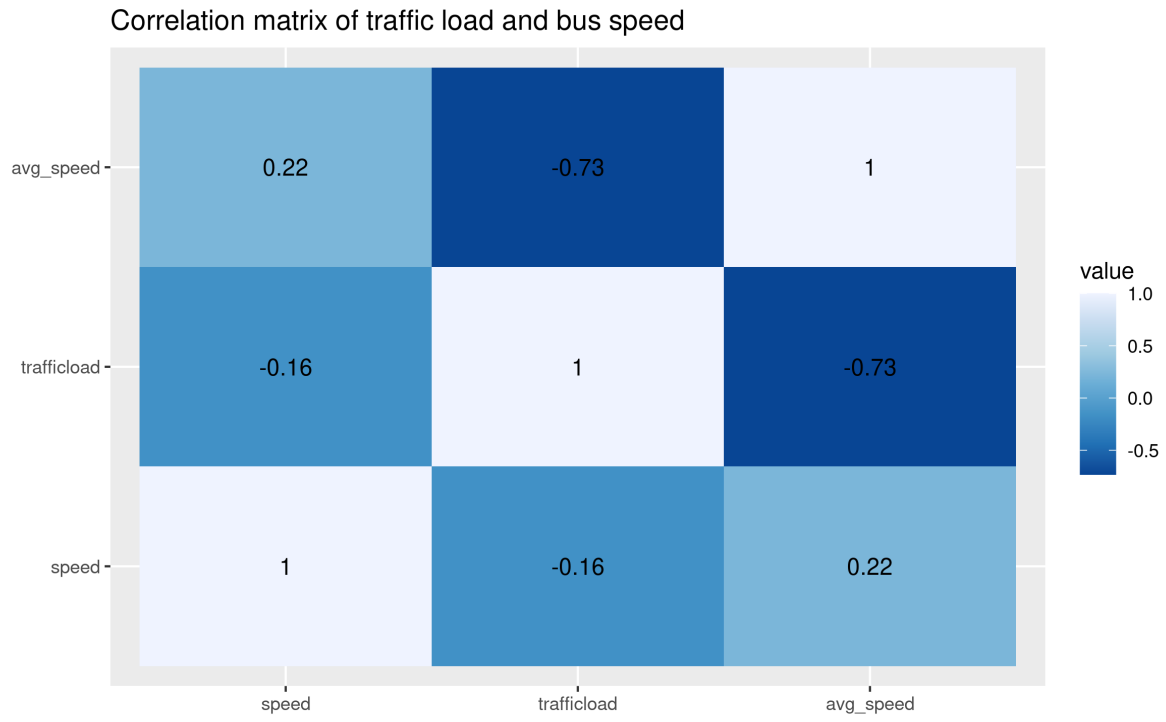

Figure 3.4 – Correlation matrix of bus commercial speed and traffic during the COVID19 1st lockdown in 2020 in France.

Our observations show that traffic has a real impact on bus commercial speed, not only by the existence of a correlation between traffic load, traffic average speed and bus commercial speed, but by showing that when there is nearly no traffic (during lockdown), the bus commercial speed tends to increase.

avg bus commercial speed	avg traffic load	avg speed on road	traffic load > 50%	traffic load = 100%
23.7 km/h	20%	42 km/h	10% of data	0.7% of data

Table 3.4 – Traffic data summary for lockdown period.

3.3.4 Ridership and fare-payment system

Nominal

We collected 20'042'000 readings for the nominal period starting from 2019-01-01 to 2020-03-16. Fig. 3.5 is the correlation matrix of the bus commercial speed and amount of smart card or ticket boardings for the nominal period. One can see that there are two very weak negative correlation (resp 0.18 for smart cards and -0.1 for tickets) for both smart card and ticket boardings. Table 3.5 shows that there are around 4 times more smart card boardings per stops in average than ticket boardings. We would expect ticket boardings to be more impacting because of ticket being sold onboard from time to time, yet smart card boardings impact on bus commercial speed looks higher here.

avg bus commercial speed	avg smart card at stop	avg ticket at stop
18.6 km/h	2.9	0.6

Table 3.5 – Ridership data summary for nominal period.

Lockdown

We collected 196'000 readings for the lockdown period starting from 2020-03-17 to 2020-04-30. Fig. 3.6 is the correlation matrix of the bus commercial speed and amount of smart card or ticket boardings for the lockdown period. It appears that the impact of smart card boardings is still visible with a smaller correlation coefficient (-0.12), accompanied with a 2 times decay in smart card boardings, as shown by Table 3.6. On the other hand the ticket boardings impact has totally disappeared, while ticket boardings have not totally disappeared as confirmed by Table 3.6. An hypothesis to explain this is that the onboard selling of tickets have been completely stopped during the lockdown, hence the tickets wearers would have necessarily bought their transport title outside of the buses. Table 3.6 also shows that the average bus commercial speed of the buses have slightly improved during the lockdown with an enhancement of around 2 km/h

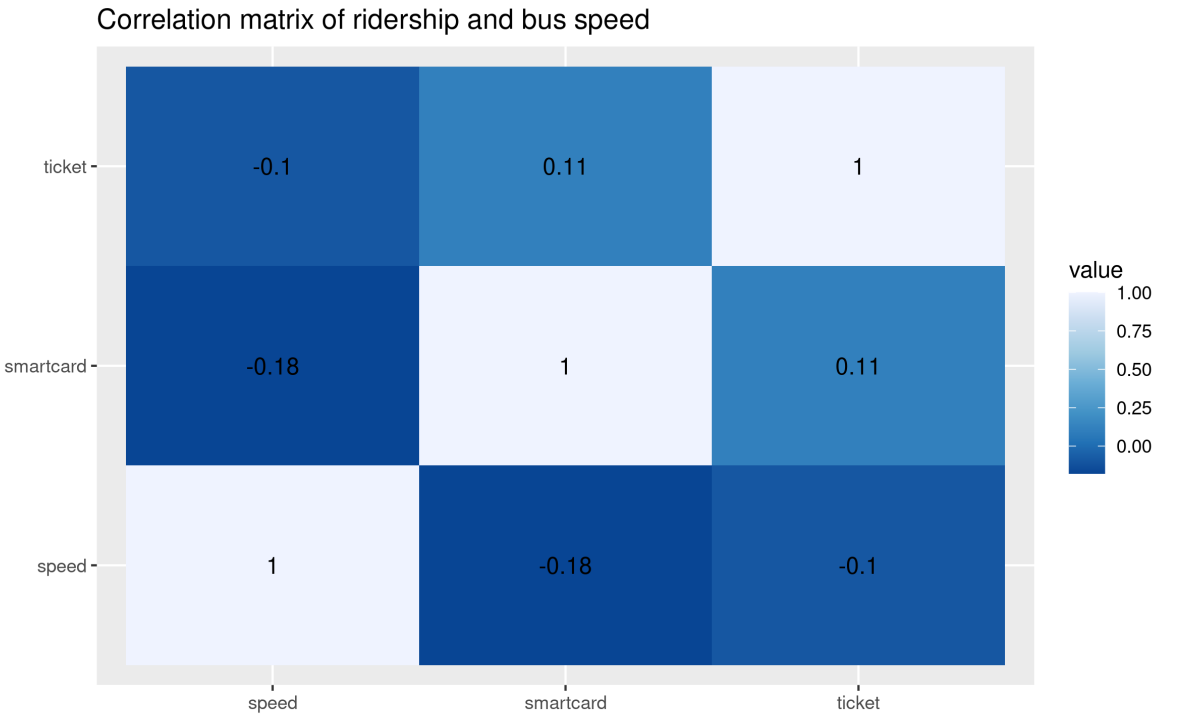

Figure 3.5 – Correlation matrix of bus commercial speed and ridership during nominal period.

avg bus commercial speed	avg smart card at stop	avg ticket at stop
20.7 km/h	1.4	0.1

Table 3.6 – Ridership data summary for lockdown period.

Our observations show that ridership has a real impact on bus commercial speed, not only there seem to be correlations between smart card boardings, ticket boardings and bus commercial speed, but moreover, when the ridership is reduced, the bus commercial speed rises.

3.4 Discussion

The previous section showed that bus commercial speed impacts are numerous and complex to gather and analyze. Yet, we suggest that we can yield a few learnings from that. Understanding and mastering the roads infrastructure is probably a good way to know how to act on the equipment, hence which of them should be avoided or favored

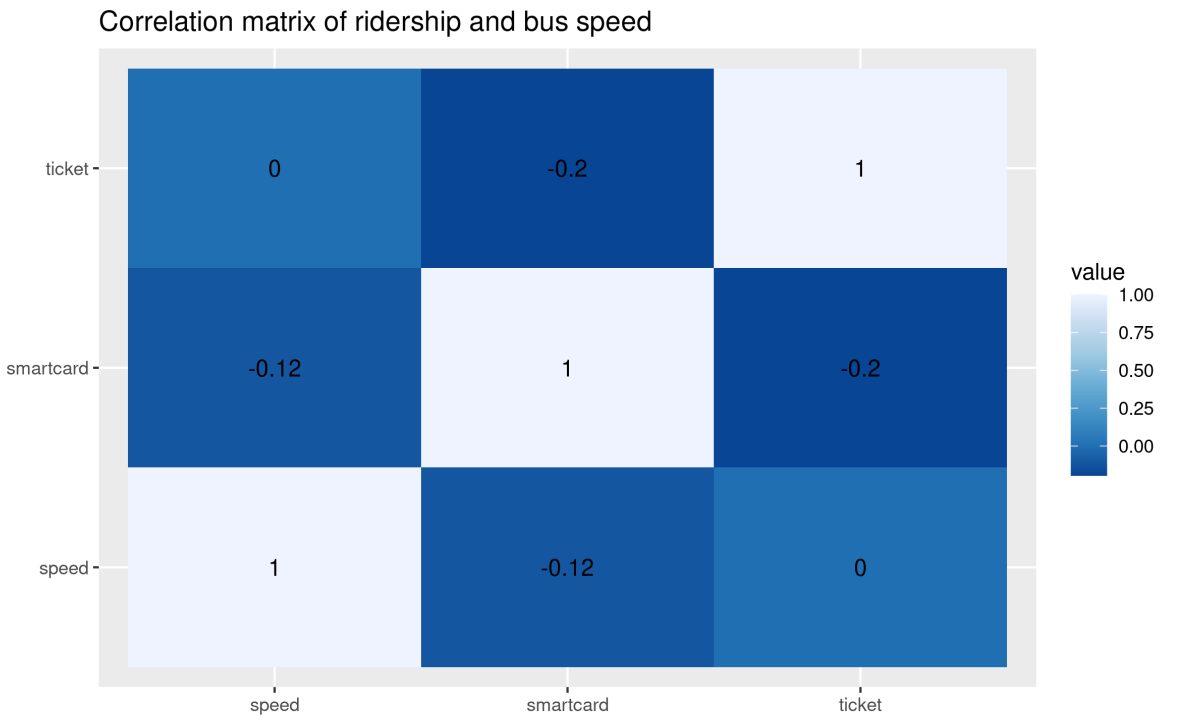

Figure 3.6 – Correlation matrix of bus commercial speed and ridership during the COVID19 1st lockdown in 2020 in France.

if one wants a high bus commercial speed, for road infrastructure is usually managed by cities themselves. A broader study, over other cities would be a good way to assess whether this suggestion is relevant or not. The tool we developed to gather infrastructure from bus network is available **here** and could help others to repeat this experiment in other areas.

The weather is probably an important factor in cities that are subject to snowfall or heavy rainfall. However, what we obtained from our data was showing that, in the particular case of Rennes, the weather impact on bus commercial speed is most of the time negligible.

Bus hardware has also been identified as a potential factor to take care of. We were able to highlight that bus commercial speed can vary if we classify readings using bus hardware and properties. However, this data have to be completed with operational information such as bus driving particularities (e.g. electrical buses / articulated buses), and the fleet distribution over the network, for buses can be dedicated to serve specific bus lines or areas only during their whole life, creating bias in hardware analysis.

Bicycles seem to have an impact on bus commercial speed, at least during hot seasons. We confirmed this assertion partially and locally, with a survey we led on a single inter-station. Thus there is a need of broader surveys to understand at what extent bicycles are impacting the bus commercial speed on various areas.

The traffic data we analyzed suggested that traffic have a real impact on bus commercial speed. We had an opportunity to compare the bus commercial speed when the bus network is in a nominal state, and in degraded state like during the first lockdown of 2020. This situation, comparable with a quasi removal of the traffic from the city, helped us to assess our hypothesis about the traffic impact on bus commercial speed. However the survey could not be led all over the network because of data availability and completeness issues, which is a risk to the generalization of the hypothesis.

Ridership was also found to have an actual impact on bus commercial speed. As for the traffic data, we attested this hypothesis by comparing the ridership impact on bus commercial speed when the bus network is in its nominal state, and when the ridership is hugely reduced, yielding a higher bus commercial speed.

All those observations helped us to raise the following **hypotheses** about the understanding of the impacting factors of bus commercial speed :

— **H1**: Road infrastructure have a major impact on bus commercial speed, both negative (traffic lights, pedestrian crossings) and positive (bus stop spacing, speed limit, roundabouts).
— **H2**: The combination of bus lane and traffic lights can result in the lowering of bus commercial speed.
— **H3**: Bicycles can have an impact on bus commercial speed when authorized on bus lanes.
— **H4**: Ridership have an impact on bus commercial speed.
— **H5**: The impacts of smart card boardings and tickets boardings on bus commercial speed are alike if tickets are sold off-board.
— **H6**: Heavy rain can have an impact on bus commercial speed.
— **H7**: Electrical buses driving specificities can yield a lower bus commercial speed.

Finally, the fact that all our analysis were made at inter-station level generates the possibility that some observations that are not visible at this scale, would be if the same analysis were led at e.g. bus line scale.

3.5 Conclusion

In this chapter we used heterogeneous datasets to analyze the impacts of factors that are supposed to vary the bus commercial speed. We led our survey on the bus network of the city of Rennes, France. We were able to test the impacts of road infrastructure, ridership, road traffic, bicycles, and weather. Our results provided evidence for the existence of these impacts, but we were unable to provide formal and definitive proof. However, the 2020 containment was an opportunity to collect data that allowed us to compare the impact of traffic and ridership when the bus network is in a nominal state, or in a degraded state with almost no traffic and ridership. The results reinforced the assumptions that traffic and ridership have a negative impact on bus commercial speed. Finally, we formulate a series of hypotheses that could help bus networks operators. However this will need them to be validated or not through larger studies in the future.

Chapter 4

Quality of compositional prediction for what-if scenarios on graphs

4.1 Introduction

In the previous chapter we focused on the factors impacting the bus commercial speed. Our findings helped the writing of this chapter, in which we study the prediction of bus commercial speed using so called micro-prediction and heterogeneous data. Indeed, knowing which data to use to feed the predictive models is a paramount step for quality predictions.

Machine learning techniques have been applied on static, graph-based applications, such as transportation networks or energy grids, to name a few [29, 4, 62]. These applications have in common an a priori topological model, i.e. a graph, where practical measures are performed on nodes and edges. In a a bus transportation network for example, it is possible nowadays to predict the awaiting time in a station or the probable duration of a trip with an acceptable accuracy [75, 68].

Recently, micro-learning approaches have been successfully applied to more dynamic, what-if scenarios (e.g. power grid management[27]). In these approaches, ad hoc models of local data are built instead of one large model on the overall data set. Micro-learning is typically useful for incremental, what-if scenarios, where one has to perform step-by-step decisions based on local properties. Going back to our bus transportation example, a typical what-if scenario applies when some roadworks (or other unforeseen condition such as flood) happens on a street, and the bus network should reconfigure itself (self-heal) by diverting the impacted bus lines. In order to find the best new routes, a straightforward algorithm is then to perform a greedy search on the next road segment, using a prediction of a bus trip duration on this segment.

This step by step, greedy approach is natural in graphs. A common feature of these graph-based applications is that the predicted properties (such as speed of a bus line) are compositions of smaller parts (e.g. the speed on each bus inter-stops along the line). But up to our knowledge, little has been written about the quality of such composite predictions using micro-models, when generalized at a larger scale.

In this chapter we propose a generic technique, graph-based compositional prediction, that allows 1) the prediction of the behaviour of composite objects, based on the predictions of their sub-parts and appropriate composition rules, and 2) the production of rich what-if analytics scenarios, where new objects never observed before can be predicted based on their simpler parts. We show that the quality of such predictions compete with macro-learning ones, while enabling what-if scenarios. We assess our work on synthetic data and a real size, operational bus network data set.

The rest of the chapter in organized as follows. In Section 4.2 we present our motivational scenario. Our model is outlined in Section 4.3. Section 4.4 shows our experiments. Finally, we present our conclusion and future work in Section 4.5.

4.2 Data Model and Motivation Scenario: Rennes City Bus Transportation

Our running example is the bus network of the French city of Rennes, forming a directed graph $G = (V, E)$, where V is the set of bus stops and $E \subseteq V \times V$ is the set of possibles one-stop trips. Such a graph can be considered as a static graph as long as most of its structure does not evolve in the short term (few months to few years). However, its edges generate a lot of data over time. We consider a non-empty set of features F, associated to each edge, with their corresponding types T_F. A typical set of features associated to each element of E is $F = (time, line, length, road - type, bicycle)$, where *time* is a timestamp, *line* is an integer denoting a bus line number, *length* is the length of the inter-stop section, $road - type$ indicates the kind of road the bus runs on (dedicated road or not), and *bicycle* indicates whether bicycles are allowed. Figure 4.1 gives an overview of a bus network graph in which green vertices are departure terminals, blue vertices are transition/departure bus stops, white vertices are transitional bus stops and red vertices are ending terminals.

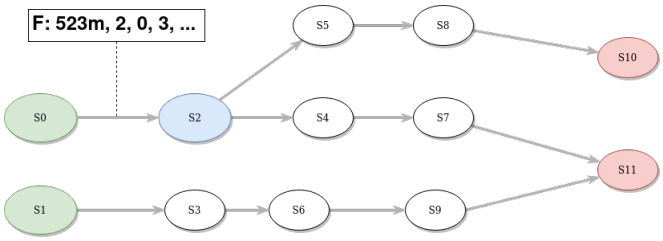

Figure 4.1 – A bus network with featured edges (F: length of the road, type of line, etc.)

Let us consider a measure $\mathcal{M}$ of the graph, defined on paths p in G. Let $\mathcal{D}$ be a learning data set of examples of $\mathcal{M}$. Let us consider a decision problem in the graph, e.g. rerouting due to roadworks. Given a source and target vertex, two approaches can be envisioned:

— **Classical prediction scenario**: enumerate possible path p from source to target and predict $\mathcal{M}(p)$ for a given timestamp, and choose the best option.
— **What-if prediction scenario**: starting from the source, make a local prediction using the micro-model on outgoing edges, choose the best option and continue in a step-by-step greedy search towards the target. The prediction on the new path p, $\mathcal{M}(p)$, is the composition of these micro-predictions. Figure 4.2 gives an example of such a task, where we want to predict the duration for a given timestamp of trip $S0 - S10$ through $S0 - S5$ and $S5 - S7$ (detours), edges that do not exist in the example data set.

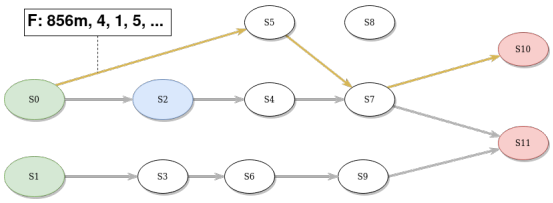

Figure 4.2 – What-if scenario: predict $S0 - S10$ while edges $S0 - S5$, $S5 - S7$ and $S7 - S10$ do not exist in the data set

In the next section we detail the corresponding learning problem associated with the what-if scenario.

4.3 Micro-learning and Compositional Prediction

As a possible way to envision the what-if scenario, we propose the notion of compositional prediction using micro-learning:

— (Micro-learning) we first learn a model of $\mathcal{M}$ on each individual edge of G.
— (Compositional prediction) Then we build the model of $\mathcal{M}$ on a new path by composition.

More precisely, let $p = (e_1, \ldots, e_n)$ be a path in the graph G. Our goal is to predict a given measure $\mathcal{M}(p)$ on any path p on this graph. We possess local measures $m(e)$ of $\mathcal{M}$ on any edge e along this path and, typically, a composition law $\mathcal{C}$ links these measures:

$$\mathcal{M}(p) = \mathcal{C}(m(e_1), \ldots, m(e_n)).$$

Consider that $\mathcal{M}$ is the trip duration for a precise bus and date, from the start to the end of the line (the path p). We have observations for $m(e)$, the duration of inter-stop trips on p for this bus. Of course in this simple case the value of $\mathcal{M}(p)$ can be obtained by summing durations of trip duration $m(e_i)$ on all edges e_i of p. Similarly, the average speed could be obtained with a slightly different composition law, weighting duration by distances. Besides additive composition laws, multiplicative laws could be foreseen, such as the probability of bus failure (which is a product of one minus the bus failure probability on each edge).

We then define our compositional prediction approach:

Compositional prediction

Let $G = (V, E)$ be a directed graph, $\mathcal{M}$ a target measure, a composition law $\mathcal{C}$ and a data set $\mathcal{D}$ of observations $m(e)$ on edges of G. Given a target path $p = (e_1, \ldots, e_n)$, the compositional prediction of $\mathcal{M}(p)$ is given by:

— building a model $m^*(e)$ to predict $m(e)$ for each edge $e \in V \times V$, according to edge features $F(e)$;

— approximating the prediction of $\mathcal{M}(p)$ by

$$\mathcal{M}^*(p) = \mathcal{C}(m^*(e_1), \ldots, m^*(e_n)).$$

Suppose that we want to estimate the end-to-end duration of trip for a new bus line in a city. Given its path p, we would learn the typical duration of existing bus lines on each inter-stops road fragments (i.e. edges), and simply sum these predictions (the composition law). If a road fragment has never been visited by any other bus in our observation data set, the model approximates it using the most similar road fragment (according to its features such as road type, number of traffic lights, and so on.)

The main advantage of this approach is that predictions are based on the underlying structure of the graph, rather that on macro-observations on it. As shown in the examples, it yields a large flexibility in predictions, without domain specific knowledge (e.g., in the bus network context, multi-agent simulation requires complex modeling hand-made tuning [43]).

But this method can also have its drawbacks. When an edge measure is missing, a strategy has to be deployed to fill the missing data, and this is highly related to the amount of describing features on the underlying graph. Also, the cost of building many models for each edge could be tremendous, regarding its macro-level counterpart. Finally, and more importantly, as each model bears its own approximations, errors may sum up along the compositional law $\mathcal{C}$, giving a potentially unusable prediction. We then identify the following research questions:

— Q1: Does micro-learning offers good accuracy, and does a rich feature set is mandatory to fill missing edge observations in the data set?
— Q2: Can usable predictions (wrt. quality) can be obtained with the compositional prediction strategy for the step-by-step, what-if scenarios, and does it compete with traditional prediction methods (that apply in different contexts)?

In the following experiments, we evaluate and discuss these questions in depth, on two sets of data. The first set comes from a synthetic bus network simulation, where timing data is 100% correct by construction. The second set is real data coming from a real bus network: this data being quite imperfect as it is most often the case for real bus networks [41, 57].

4.4 Experiments

4.4.1 Experimental settings

Reproducibility

All of the experiments are reproducible with code and resources available on a dedicated repository [1].

Experimental environment

All experiments were run on a 16 cores 32 thread @ 3500MHz, 64GB DDR4 3200MHz 1TB SSD NVMe, running Ubuntu 20.04 (Budgie flavour). Data were gathered into a normalized data lake built upon Apache Spark 2.4.5/Scala 2.11.12 jobs. All learning tasks were performed using GraalVM 20.2 and Scala SMILE 2.5.3. It took around 70 minutes for overall model training and 2 minutes for each of the following predictions.

Experiment on Synthetic Data

We created a data synthetizer that generates an artificial bus network based on real world features evenly distributed over it using uniform random generation. Using this code, we created an artificial bus network made of 2000 inter-stops (edges) representing 110 different bus lines, each made of 12 to 43 inter-stops. Hence some inter-stations are shared between bus lines. Note that we do not need to generate nodes as long as nodes features are not used in our model which is edge based (nodes are virtually represented by the fact that edges are connected to each other within bus lines).

We then generated 25 to 150 virtual bus trips for each of those lines, generating a target measure for each edge used in the trip. We used a negative exponential based function that takes edges features into account (a typical modeling in bus networks).

$$V_c = (V_0 + V_0'\delta D + V_0''\delta P + V_0'''\delta L)e^{-[(\alpha+\alpha'\delta fp)fp+(\beta+\beta'\delta tp)tp+(\gamma+\gamma'\delta fs)fs+(\sigma+\sigma'\delta ts)ts]} \quad (4.1)$$

Equation 4.1 represents the commercial speed equation proposed by [23]. Because of the efficacy of this equation, and the fact that it is easy to tune to make it close enough to

1. https://gitlab.inria.fr/glyan/compred

the reality, we base our data generation equations on it, yielding **Equations 4.2 and 4.3** .

Table **4.1** Exposes the variables we use in the equations, and the range of the value they can randomly take. Each coefficient in the equations are manually tweaked in order to obtain values that look correct to us. Once done, and for each scenario (travel time and risk), we generate a value μ using the corresponding equation and apply it as the mean of a normal law for which we generate a random standard deviation σ, corresponding to a random value situated between 2.5 and 15% of the mean μ we generated earlier. We then use this normal law $normal(\mu, \sigma)$ to generate random values for our datasets.

$$TravelTime = 35 * e^{-(0.03*d+0.02*h+0.05*p+0.0002*l+0.002*s+0.001*c+0.0002*g+ 0.0003*ra+0.007*st+0.01*ts+0.0002*pb-0.0001*pc+0.0001*po+0.0002*ps)} \quad (4.2)$$

$$Risk = 1 * e^{-(0.00002*d+0.00002*h+0.000035*p+0.000002*l+0.000005*s+0.000003*c+ 0.000005*g+0.000003*ra+0.000002*st+0.000005*ts+0.000002*pb-0.000001*pc+0.000001*po+0.000002*ps)} \quad (4.3)$$

Variable	Meaning
d	day of the week $\in [0, 6]$
h	holiday $\in [0, 1]$
p	period in day $\in [0, 2]$ with 0=free, 1=avg traffic, 2=rush hour
l	length of the inter-stop in meters $\in [150, 600]$
s	maximum speed $\in [30, 55]$
c	number of crossings $\in [0, 4]$
g	number of giveways $\in [0, 3]$
ra	number of roundabouts $\in [0, 2]$
st	number of stop signals $\in [0, 1]$
ts	number of traffic signals $\in [0, 3]$
pb	proportion of road shared with bikes $\in [0.0, 1.0]$
pc	proportion of bus corridor $\in [0.0, 1.0]$
po	proportion of one way road $\in [0.0, 1.0]$
ps	proportion of slow zone $\in [0.0, 1.0]$

Table 4.1 – Variables of our equation used for data generation

Doing so we generated two datasets: one for which measure is the travel time in seconds. Hence its compositional result is obtained by summing the travel time of each

edge; another one for which measure is inter-stop reliability (i.e a real value between 0 and 1,. 0 being unreliable and 1 totally reliable), with its compositional rule being made of product of edge reliability.

For each of those datasets, we applied a 30-fold cross-validation. For each fold, all of the data of one random bus line is removed from the training dataset, and is then used for testing.

Experiment with Real Data

We considered the bus transportation system of the French city of Rennes and its metropolitan area (about 500,000 inhabitants) managed by the Keolis Rennes company. Our graph is the graph of 2110 bus stops and 119 bus lines [2], and the features of road portions are automatically extracted from OpenStreetMap (OSM in the sequel).

More specifically, the raw historical data set is composed of readings of bus travels between two contiguous bus stops. Table 4.3 shows a sample of this data set. The major columns are *line_type* (0 being high frequency, 1 urban, 2 metropolitan and 3 express metropolitan lines); *dwell_time*, the time passed at the origin stop (taking and unboarding passengers) and *speed*, the arithmetic speed between *orig* and *dest*. A single trip, which consists of all the measures of a single bus passing through every single bus stop of a bus line, can be identified by combining the columns *trip_start* which is the instant at which the bus has left the first point of the bus line, *sm* which is the service identifier, *chaining* which is the version of the bus line and *dir* which is the line direction. We considered only real data: no imputation was made in our data set. We considered only full trips for which all data points were present. We also filtered historical data used to feed the model, dropping invalid speeds/nulls/values and outliers. Using these rules, **we gathered 12 month of data between January 2019 and December 2019, for a total of around 15 millions tuples [3].**

Prediction task

We targeted the prediction of **the speed of a bus line** depending on time (holidays period, day, time of the day) and line type (urban bus line, metropolitan bus line). We considered predictions for existing lines (classical prediction) and non-existing lines (what-if scenarios, where the lines and some inter-stops data are absent in the training data set).

2. Rennes transportation network: https://data.rennesmetropole.fr
3. The data and more material is available at https://gitlab.inria.fr/glyan/compred

Candidates

As experimental candidates, we chose 4 groups of bus lines for which we could gather data all along their path. We tested 13 different bus lines in total. The four groups contain different class of bus lines: urbans, inter-district, metropolitans and express.The urban group is composed of urbans bus lines, that cross the city center along their path. The inter-district group are urban bus lines that link different districts, avoiding the city center. The metropolitan group is composed of metropolitan bus lines, that links cities of the metropolitan area to the main city. Those lines contain urban, peri-urban and metropolitan sections. Finally, the express group contain metropolitan lines that allows a faster travel from and to external cities by servicing less bus stops. Note that in the network, each line share some sub-paths with others. Table 4.2 shows the specifications of each candidate bus line. For each bus line, its type is exposed (resp. urban/inter-district/metropolitan/express) along with their identifier and chaining (variant identifier) and direction. Other important data such as the number of inter-stations that compose the bus line, the inter-stations that are not shared with other bus lines (noted as nb_unknown_inter-stations) are also visible. Finally, the amount of candidate trips for prediction test for compositional and macro models are shown.

Table 4.2 – Bus lines metadata

type	id	chaining	direction	nb_inter-stations	nb_unknown inter-stations	nb_trips_Compred	nb_trips_macro
urban	858	1	R	40	16	3795	3794
urban	787	42	A	32	3	617	1352
urban	785	20	R	40	24	2207	2205
urban	889	5	A	28	5	3959	3960
urban	689	18	A	36	2	1961	1961
inter-district	696	1	A	33	11	1208	1209
inter-district	581	2	R	28	7	46	48
metropolitan	683	55	R	20	11	2472	2472
metropolitan	849	72	A	30	1	777	778
metropolitan	532	23	R	19	1	1689	1685
metropolitan	969	21	R	11	1	869	872
express	711	15	R	6	1	522	522
express	859	10	R	7	2	299	286

trip start	line	line type	sm	dir	orig	dest	speed	dwell time	order orig	order dest	section	chaining
2018-07-02 00:05:00	02	0	0221	A	2844	2842	25.8	0	1	2	1	21
2018-07-02 00:05:00	02	0	0221	A	2842	2804	21.0	23	2	3	1	21

Table 4.3 – Sample of the historical data set

Road features with OpenStreetMap

We have built a tool to look for a bus network in OpenStreetMap and to extract its infrastructure information, road per road. We then aggregate the data grouping the roads

period	day	holidays	line type	real speed	dwell-time
1	0	2	0	32.1	10
4	0	5	0	23.8	8

legal speed	length	nb traf. signals	nb crossings	nb stops	nb giveways	nb lvl crossings	nb roundabouts
50.0	507.5	0	4	0	0	0	1
50.0	584.5	0	8	0	3	0	2

proportion bus lane	proportion bike on road	proportion one way	proportion slow zone
0.72	0.85	0.87	0.0
0.0	0.11	0.68	0.0

Table 4.4 – Sample of the training data set used for micro-learning

by inter-stops. We then sum the road lengths, number of traffic signals, stops, giveways, roundabouts, level crossings, crossings, and add the average legal speed of the roads.

Time discretization

We categorized the time periods into six categories: No holidays, and Autumn/ Christmas/Winter/ Spring/ Summer holidays. For each category we divided each day coded from 0 (monday) to 6 (sunday) in 1 to 3-hours period (e.g. 1am-3am, or 7am-8am). This division is due to obvious pace changes in bus usage, as confirmed by the Keolis company.

Micro-learning data set

The micro learning data set is made of measures for each inter-stop of the network. We simply identified every distinct inter-stop and added meta data such as holidays, bus line type, OSM data (that is constant through time), period of time in day and week day (Table 4.4).

Macro-learning data set

The macro learning data set is built using the raw data set, from which we extract **only the overall speed of every complete trip** (i.e trips for which we have data for each inter-stop) of every bus line. We then add the bus line, its type and finally the OSM data aggregated all along the line (i.e total number of traffic signals, stops, etc.). In order to keep as much data as possible, we did not use traffic and ridership data due to the fact that they cover smaller areas and periods in time.

Learning method and validation methodology

Our models are built using a classical random forest algorithm parameterized for regression and optimised using grid-search (this choice is motivated by its generality and overall performance, after selecting it amongst others: Lasso, OLS ,SVR, Cart, bayesian ridge and gradient boosting, testing their performance using a small sample of data. It could be naturally adapted to more specific methods, but this is not our target here). For micro-learning, we build a model of inter-stop speed. We then predict the speed of any trip by the natural compositional prediction $\mathcal{C}_s$ (summing the road lengths divided by the summed travel times, obtained with the predicted speeds). For macro-learning, we build a model of full bus-line speeds (from start to stop). Learning is performed with OpenStreetMap features, and without OpenStreetMap features on every bus lines in order to assess the impact of these features on the model precision. For validation purposes, we have removed all the data of candidates bus lines from the training data set of each model. To evaluate the classical scenario, we predicted the speed of an entire line with macro and compositional prediction. For the what-if scenario, we measured the compositional prediction quality on non-existing paths in the data set.

In the next sections we are coming back to our initial questions from Section 4.3.

4.4.2 Micro-learning Accuracy (Q1)

The experiment we led using artificial data yielded the following results:

Prediction type	RMSE	MAE	MAPE	Average part of known inter-stops
Micro	4.78s	0.12s	7.49%	69.04%
Macro (Entire bus line)	62.36s	51.96s	3.76%	69.04%
Compred (Composition over entire bus line)	25.36s	3.23s	1.75%	69.04%

Table 4.5 – Prediction accuracy on a synthetic dataset for additive composition law (bus line duration)

Prediction type	RMSE	MAE	MAPE	Average part of known inter-stops
Micro	0.019	0.001	1.50%	65.17%
Macro (Entire bus line)	0.073	0.012	7.69%	65.17%
Compred (Composition over entire bus line)	0.072	0.006	7.63%	65.17%

Table 4.6 – Prediction accuracy on a synthetic dataset for multiplicative compositional law (product of probabilities)

Tables 4.5 and 4.6 show that predictions for our synthetic datasets, the accuracy of our micro-model on edges is quite good for both speed and reliability prediction. But how

would it behave with real, imperfect data that is typical of bus networks?

Accuracy We tested (predicted) ∼12'500 trips for the urban bus lines group, ∼1'250 trips for the inter-district group, ∼5'800 for the metropolitan group and ∼800 trips for the express group. All those trips are distributed over the 6 holidays regime, 7 days and 17 periods in day over the 2019 year.

Figure 4.3 shows the predicted speed of one of the urban bus lines using micro-learning. Each point represents the average predicted speed (purple) and average real speed (dashed) of each inter-station of the line on all the time periods for which data was available. The leftest point of each curve represents the beginning of the bus line, the rightest one being the last stop. Observe that due to the bus line's data deletion from the training dataset, some inter-stops that are specific to this line are totally unknown to the model (orange dots). We also considered metropolitan, express and inter-district bus lines (not displayed) [4]. Visually at first, we can see that micro-learning captures well the daily behaviour of buses, even when the inter-stops are unknown to the model (orange dots).

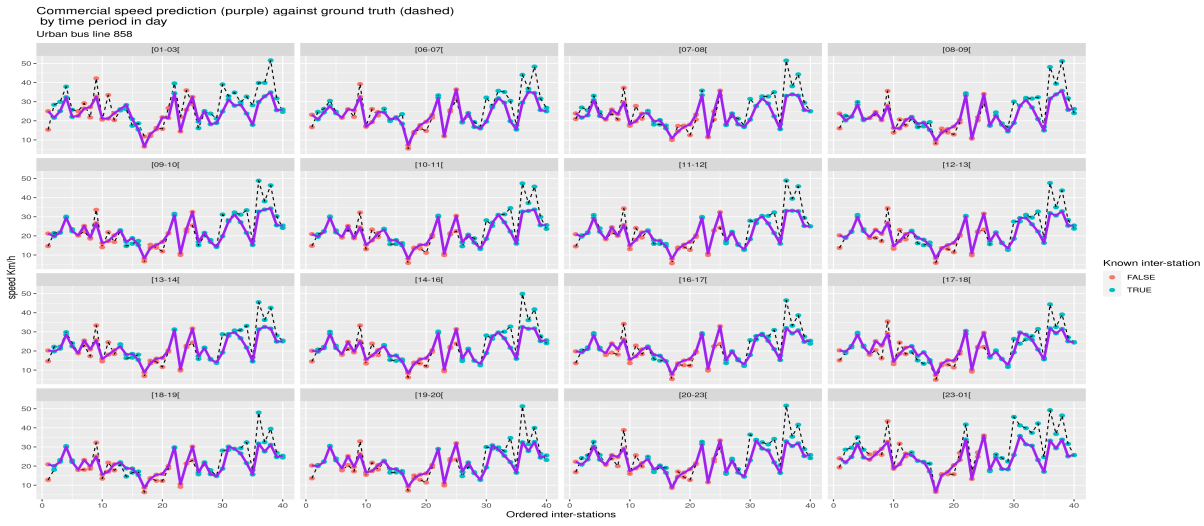

Figure 4.3 – Speed of urban bus line 858 at inter-stations using micro-learning (purple) vs true speed (dashed). Orange inter-stations where not used by any other bus, i.e. are absent from the training set.

Table 4.7 sums up the models's prediction errors for each bus line, group of lines (urban, inter-district, metropolitan and express) and all the lines (global). For each model and bus line (resp. group of lines), the table presents the Root Mean Squared Error (RMSE),

4. More material is available in the public deposit

the Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE). RMSE emphasizes large residuals (outliers) while MAE and MAPE are less sensitive to residuals and easier to interpret [54]. RMSE and MAE unit is km/h while MAPE's unit is percentage (difference against truth in percents).

Table 4.7 – Results table

Line	Micro known			Micro unknown			Micro global			Compred			Macro		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Urban lines															
858	7.3	5.0	18.2%	6.1	4.5	25.9%	6.9	4.8	22.4%	1.6	1.2	6.2%	2.1	2.0	8.4%
787	5.6	3.9	19.9%	9.1	7.3	44.4%	6.0	4.2	22.2%	1.7	1.3	7.2%	2.2	2.0	9.7%
785	5.9	4.5	22.7%	8.3	6.0	29.2%	7.5	5.4	26.6%	2.6	2.0	9.9%	2.7	2.5	10.4%
889	5.6	4.3	31.1%	8.7	5.9	25.4%	6.3	4.6	30.1%	3.0	2.7	17.4%	2.7	2.6	15.4%
689	6.3	4.6	29.4%	8.2	6.7	53.9%	6.4	4.7	30.7%	3.2	2.8	16.5%	2.4	2.3	12.2%
All urbans	6.3	4.6	25.2%	7.5	5.4	29.4%	6.7	4.8	26.5%	2.6	2.1	12.0%	2.4	2.3	11.5%
Inter-District lines															
696	8.9	6.4	27.1%	9.3	7.4	46.2%	9.0	6.7	33.5%	3.2	2.8	13.7%	4.0	3.8	15.5%
581	6.4	5.1	26.7%	10.3	7.6	31.8%	7.6	5.7	27.9%	3.5	2.8	12.3%	3.9	3.5	13.4%
All inter-Districts	8.8	6.4	27.0%	9.3	7.4	45.9%	9.0	6.7	33.3%	3.2	2.8	13.7%	4.0	3.7	15.5%
Metropolitan lines															
683	5.0	3.8	15.5%	8.4	6.7	26.9%	7.1	5.4	21.8%	3.2	2.5	10.3%	3.1	2.9	9.5%
849	6.3	4.7	37.4%	5.0	4.0	11.9%	6.3	4.7	36.6%	3.2	2.8	12.1%	3.7	3.5	14.3%
532	7.9	5.7	24.2%	12.1	11.9	427%	8.2	6.0	45.4%	4.0	3.3	13.7%	4.8	4.5	15.5%
969	7.2	4.8	25.3%	6.0	4.4	16.1%	7.1	4.8	24.4%	3.6	2.6	9.8%	3.5	3.2	9.9%
All metropolitans	6.7	4.8	25.6%	8.6	6.8	48.3%	7.3	5.4	31.7%	3.5	2.8	11.4%	3.8	3.5	11.9%
Express lines															
711	5.8	4.2	36.6%	4.5	3.6	10.1%	5.6	4.1	32.2%	3.6	2.7	10.3%	7.0	6.9	21.1%
859	6.1	4.2	27.8%	7.9	5.5	21.9%	6.6	4.6	26.1%	7.3	5.6	24.7%	7.0	6.8	20.2%
All express	5.9	4.2	33.4%	6.5	4.6	16.4%	6.1	4.3	29.8%	5.3	3.8	15.6%	7.0	6.8	20.8%
All lines															
All lines	6.6	4.7	25.5%	5.7	7.8	33.7%	7.0	5.0	28.0%	3.0	2.4	12.1%	3.2	2.9	12.2%

group	Compred standard dev.	macro standard dev.
urban	1.0	0.5
inter-district	0.7	0.3
metropolitan	0.9	0.4
express	0.9	0.0
global	0.9	0.4

Table 4.8 – Models's sensitivity to time features

If we take a look at micro predictions in Table 4.7, we observe that prediction errors for known inter-stations (classical prediction) are better than prediction errors of unknown inter-stations (what-if predictions), except for the express bus lines group. Indeed, RMSE, MAE and MAPE can vary from a 1.2 to 2 factor between micro known and unknown. This shows that the features set used for micro learning might need some enhancement to help the model predicting better. Finally, we observe that global micro predictions precision vary between bus lines groups. However, the urban group gets the best predictions in all aspects when compared to other groups (RMSE:6.7, MAE:4.8, MAPE:26.5%), followed by express (RMSE:6.1, MAE:4.3, MAPE:29.8%), metropolitan (RMSE:7.3, MAE:5.4, MAPE:31.7%) and inter-district (RMSE:9.0, MAE:6.7, MAPE:33.3%) groups.

Urban bus lines are the major lines in the network, hence they produce a significant part of the data used for training. This can explain why the models are more precise when they predict speed for this group.

We noticed a huge prediction error for metropolitan bus line 532 with resp RMSE, MAE and MAPE at 12.1, 11.9 and 427%. We investigated how can the model perform that bad on some cases and found that the average speed of the only unknown inter-station of the bus line 532 (cf. Table 4.2) is lower than 5km/h, while its features are alike others inter-stations in the network. The speed calculation business rules used might yield such a low speed, e.g if the bus has to wait at a bus stop for operational reasons. In other words, inter-stations that have very low average speed with non specific features set is prone to yield important predictions errors (i.e considered as an outlier).

Globally, micro-learning yields variable quality results, with a higher error rate for unknown inter-stops. The features used for predictions probably need enhancement in terms of noise reduction and/or external data addition such as smart-card data, traffic status, etc.

Sensitivity with respect to features We checked the models sensitivity to OpenStreetMaps features and time features by training models with OSM-less datasets. Table 4.9 shows the results of the predictions made without OSM features, hence using time discretization and built-in features only (period, day, holidays, line type, dwell-time). Note that black, green and red values are resp. identical results, better results and worse results than their counterparts in Table 4.7. We observed that, in most of the cases (i.e individual bus lines and groups), OSM features play an important role for speed prediction for micro, compositional and macro prediction. It seems that micro prediction has to be done using OSM features [5]. Indeed, the error is often 1.5 or more than 2 times larger when compared to results of Table 4.7. Table 4.10 shows the speed prediction variation of micro-learning model with and without OSM features. Clearly, the micro-learning model trained with them is way more capable of predicting different speed for different inter-stations, with a standard-deviation of predicted speeds that is nearly three times higher than the micro-learning model trained without OSM features. However, predictions for unknown inter-stations of lines 787 and 683 are way better without OSM features than with it. Moreover, there are cases in which the removing of OSM features delivers better results for compositional and macro prediction, and not by a margin (more than 5%). This is the case for metroplitans bus lines and inter-district bus lines (Compred mostly).

Conclusion Micro-learning on graphs at edge level (e.g. bus-inter-stops level) yields very good results on synthetic data. The accuracy turns out to be variable on a real data set. Prediction error tends to be more important when no observations on this very edge is available in the learning data set. These results point the importance of choosing the right features set for the model. However, Figure 4.3 shows that the micro learning model has a sensitivity to current features, hence this allows for what-if scenarios, when one wants to predict the behaviour of a new unseen path.

If we understand that the addition of OSM features in the training datasets helps the models to link the time features and speeds to the network topology, thereby enhancing the predictions precision, an effort has yet to be made to explain the reasons of the enhancement observed on some bus lines's predictions when removing OSM features.

5. Charts of micro-learning results with and without OSM features are available in the public repository for a more visual feedback

Table 4.9 – Results table of models trained without OpenStreetMaps features

Line	Micro known			Micro unknown			Micro global			Compred			Macro		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Urban lines															
858	10.1	7.5	29.6%	8.0	6.3	49.7%	9.3	7.1	37.7%	2.0	1.6	8.6%	2.3	2.2	9.8%
787	7.3	5.6	32.8%	6.5	5.3	31.9%	7.2	5.5	32.7%	2.1	1.7	9.2%	2.3	2.0	9.1%
785	8.0	6.2	34.3%	10.2	7.6	37.7%	9.4	7.1	36.3%	2.2	1.6	7.8%	4.0	3.8	15.5%
889	7.3	6.0	51.5%	8.5	6.5	30.0%	7.5	6.1	47.6%	5.8	5.5	35.0%	4.2	4.1	25.2%
689	8.5	6.5	46.5%	8.8	7.7	76.0%	8.5	6.6	48.1%	3.9	3.7	21.1%	1.5	1.4	6.8%
All urbans	8.6	6.6	40.7%	9.0	6.9	42.9%	8.7	6.7	41.3%	3.9	3.2	18.8%	3.2	2.9	14.8%
Inter-District lines															
696	11.4	8.6	35.2%	12.0	10.1	69.7%	11.6	9.1	46.7%	2.6	2.2	10.8%	3.8	3.5	14.6%
581	8.5	6.9	36.0%	14.7	10.6	36.1%	10.4	7.9	36.0%	2.6	2.1	8.6%	7.4	7.0	26.6%
All inter-Districts	11.3	8.6	35.2%	12.1	10.1	68.9%	11.6	9.1	46.3%	2.6	2.2	10.7%	4.0	3.7	15.1%
Metropolitan lines															
683	9.1	6.9	25.3%	6.1	4.9	20.6%	7.6	5.8	22.7%	2.3	1.7	6.9%	3.2	3.0	9.8%
849	11.3	8.8	107%	8.5	7.4	21.5%	8.7	11.3	104%	3.9	3.5	15.0%	2.0	1.8	6.8%
532	13.8	10.9	49.0%	24.0	23.8	838%	14.6	11.6	90.4%	2.5	1.9	7.8%	3.0	2.8	9.1%
969	14.4	12.0	68.5%	8.5	6.4	20.4%	13.9	11.5	64.1%	4.6	3.9	12.8%	6.4	6.0	18.2%
All metropolitans	12.1	9.4	60.3%	8.3	6.0	65.7%	11.3	8.5	61.8%	3.1	2.4	9.1%	3.7	3.2	10.5%
Express lines															
711	10.0	8.7	69.1%	10.0	9.3	26.1%	10.0	8.8	61.9%	5.3	4.6	15.7%	7.2	7.0	21.7%
859	8.9	7.6	50.4%	9.7	7.9	25.0%	9.1	7.7	43.3%	6.4	5.3	19.6%	7.9	7.6	22.8%
All express	9.6	8.3	62.2%	9.8	8.6	25.5%	9.6	8.4	54.4%	5.7	4.9	17.1%	7.5	7.2	22.1%
All lines															
All lines	9.6	7.3	44.5%	9.1	7.0	48.5%	9.5	7.2	45.7%	3.7	3.0	15.5%	3.6	3.2	13.9%

group	prediction variation
with OSM features	7.5 km/h
without OSM features	2.8 km/h

Table 4.10 – Micro-learning models's sensitivity to OSM features

4.4.3 Compositional prediction efficiency (Compred) (Q2)

Comparison with macro predictions

Comparing compositional prediction results (Compred) with Macro prediction, raised the following remarks:

For synthetic data, Tables 4.5 and 4.6 show that the compositional prediction (Compred) performs at least as well as macro prediction (i.e over whole line), and quite often much better. Hence we can hope that this trend will be the same for real world data

For urban group, Compred MAPE and RMSE differences are only of 0.5% and 0.2 km/h while Compred's MAE is 0.2 km/h better than macro's. That is to say, macro and Compred are nearly on par when it comes to predict urban bus speed. In other groups, compositional prediction performs slightly better than macro predictions, with MAE, RMSE and MAPE lower for Compred than macro by around 2.5 to +25% depending on the measure we compare. Hence, this suggests that compositional prediction seems to be more capable of catching fine grain variations (i.e variation on inter-stations) than macro does, as shown in Table 4.8. This table shows the average predicted speed variation (standard deviation in km/h) of bus lines over time (holidays, days, period). We observed that Compred model seems to be at least twice as sensitive to time features as macro model. Hence, it confirms the claim we made in section 4.4.2

Conclusion To predict a measure on a path, we applied compositional prediction (here, a weighted sum of the predicted measure at the edge level). It appears that the obtained quality is good with respect to the macro-learning approach both on artificial and real world data. Actually our method reaches state of the art performance if we compare with [72]. However, real world data results are not as good as artificial ones. We interpret this as follows: micro-learning is done for all edges, and glitches in observations may perturb all these levels. Hence, the composition may aggregate all these errors. Conversely, in macro-learning, fewer models are computed, which is less error prone. But again, macro-learning does not allow the prediction of unknown path of any length, while compositional prediction can.

4.4.4 Threats to validity

As internal threats, we acquired the real data set ourselves using our own code, which may be prone to errors. But this data are used in production by the Keolis company, and has been controlled using business rules many times. Also, we used only complete data. No imputation was done. Our code uses a high-level language and state-of-the art libraries for data extraction and machine learning. As construct validity, we choose to use Random forests as ensemble methods model for regression. The choice we made was based on performance amongst a set of models tested on a small sample of data from which random forests performed better (in terms of results and computing time). However, we assumed that the models would behave in an analogous manner with a wider dataset, whereas there is a small risk that this is not the case. Finally, the scope of this chapter is to assess the quality of compositional prediction for composite objects in complex environment against traditional macro approach, we then considered the machine-learning model selection as a secondary issue, hence we probably could have different results with other models. Yet, choosing better models for compositional prediction is another issue that probably needs a dedicated work. As an external threat, our data set, time frame and targets selection might have specificities that could prevent generalization beyond these lines or Rennes Metropolis. On the other side, we gathered 15M tuples over a year from a large city common bus transportation system. The chosen lines are typical and of different kind (urban, inter-district, metropolitan and express).

4.5 Conclusion and Future work

In this work, we presented compositional prediction on graphs, an approach to infer path properties from edge properties obtained by micro-learning. Based on a real-size application, we evaluated this approach with respect to classical macro-learning. We showed that is allows for prediction on an unknown path, enabling the study of what-if scenarios. This approach exhibits at least comparable and often better quality than the classical approach, while offering what-if capabilities. Finally, the importance of a rich feature set is underlined. As a future work, we would like to finely understand the quality gap between macro-learning and compositional prediction: while good on small scales, it could be reasonable to switch to macro-prediction on relevant sub-path of the graphs. Also our model must be tested on other scenarios like predicting the fuel consumption along a line, or with other composition methods such as product e.g.; computing the risk of accident

along a path given its sub-parts individual risks. Eventually given the overestimation of the model one would try to apply arbitrated ensemble methods [12] to re-qualify the model output and enhance its precision. Finally, since the presented model could be embedded within a data exploration model, this work can be seen as a first step toward declarative languages for what-if scenarios.

Chapter 5

DataTime: a Framework to smoothly Integrate Past, Present and Future into Models

5.1 Introduction

The previous chapters can be seen as preconditions to this one. Indeed, we had to be able to gather quality data (Chapter 2), identify what data to collect in order to feed predictive models (Chapter 3), and to build predictive models that could be used at any point in space and time in a directed graph (Chapter 4). This chapter is the direct outcome of the previous ones, since it makes use of the knowledge we acquired through them.

As seen in Chapter 1, so called Intelligent Public Transportation Systems (IPTS) are complex socio-technical systems, involving people (*e.g.,* the users of the networks) as well as supporting infrastructures, from the transportation means themselves (*e.g.,* buses) to the IT supporting them [2, 56]. One key feature of IPTSs are their information systems allowing a network operator to plan, analyze and manage the network with respect to metrics such as transportation time (or commercial speed) (cf. 1.2), energy consumption or accident rates. When an IPTS at least partially relies on buses, these metrics are difficult to predict. Traffic indeed varies widely during the day, with rush hours further slowing down bus loading and unloading and compromising planned connections, with a significant impact on travel time, as seen in chapter 3. Furthermore, when some roadworks (or other unforeseen condition such as flood) happen on a street, the IPTS should be reconfigured by diverting the impacted lines using the best possible new routes, which is not an easy task for many European cities, built around centuries old, crowded and tortuous city centers.

Supporting IT systems for IPTSs are typically made of two relatively independent parts, one organized around *space* and the second one around *time*. The spatial one is an a

priori model of the network: what are the network topology (modeled as a directed graph), the transportation means (e.g., bus, tramways, metros), their itineraries, the infrastructure (e.g., kinds of roads), the scheduled departures, etc. The temporal one is made of the (huge) time series of data gathered from the many sensors available in the IPTS. The existing time series give the opportunity to not only provide a *model at run-time* [47], but also a full-fledged *digital twin* [9] of the IPTS to analyze, plan and manage the operations using machine learning techniques.

However these space and time parts of the supporting IT are most often not well integrated, making it hard in practice for network operators to leverage the avalanche of data gathered from the IPTS. For instance diverted lines (i.e., space modification) have no historical data to learn from, so even machine learning techniques that have successfully been applied on graph-based structures typical of an IPTS [29, 4, 62] cannot work out of the box.

Moreover, as for many other network problems (e.g., electricity or water networks), properties of interest (e.g., commercial speed of a bus along a line) are compositions of smaller independent parts (e.g. the speed on each bus inter-stops along the line). This enables the prediction on the behaviour of composite objects (a path in the graph), based on the predictions of their sub-parts (i.e. on edges) and relevant composition laws. This opens what-if analytics scenarios, where new objects never observed before can be predicted based on their simpler parts, and can then be used e.g., to optimize diversions in case of unforeseen events such as roadworks or floods.

A possible direction for integrating these models (spatial, temporal and predictive) would be to articulate them around the notion of digital twin and the concept of time, i.e., digital twins extending themselves towards Past, Present, and Future [38]. For that purpose, we propose a new framework, called DataTime, to implement models at run-time which capture the state of the system according to both time and space, here modeled as a directed graph. In this graph, both nodes and edges bear local and independent states (ie. values of properties of interest). DataTime offers a unifying interface to query the past, present and future (predicted) states of the system. This unifying interface provides i) an optimized structure of the time series that capture the past states of the system, possibly evolving over time, ii) the ability to get the last available value provided by the system's sensors, and iii) a continuous micro-learning over graph edges of a predictive model to make it possible to query future states, either locally or more globally, thanks to a composition law. We apply our framework in the context of an urban transportation

system, and concretely deploy and evaluate it on the IPTS of the city of Rennes (France), that is operated by the Keolis company.

The rest of this chapter is organized as follows. We first introduce the DataTime framework (Section 2), and then we present its use in the context of an IPTS (Section 3). Then in Section 4 we describe how it was deployed at Keolis Rennes in the context of the real bus network of the city of Rennes (France), and what lessons we have learned from this experimentation. Section 5 discusses related work, before we conclude and raise several perspectives in Section 6.

5.2 The DataTime Framework

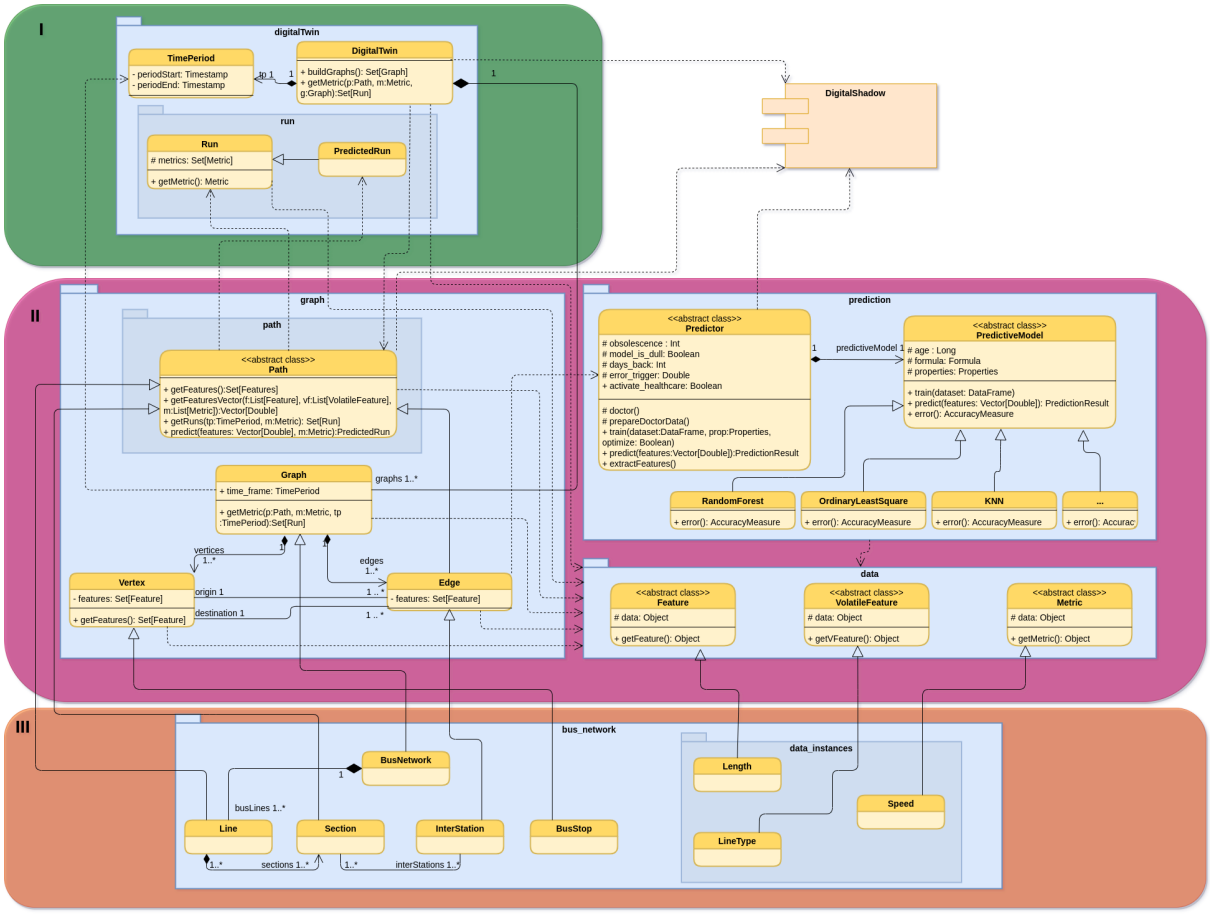

Figure 5.1 – The DataTime framework (excerpt)

Figure 5.1 shows a simplified class diagram of DataTime. The objective of this frame-

work is to enable graph-based seamless space and time exploration, through the analysis of historical data, real time data and predicted data. It is composed of two main parts: i) the spatial model and predictors configuration (part II of Fig. 5.1) that is dedicated to the designers who implement adaptations of I and II for the end-users and ii) the digital twin/shadow to reason over time about the spatial model (part I of Fig. 5.1), thus the digital twin built by the designers is the end-users' entry point to manipulate DataTime. A third component has been added in the part III of Fig. 5.1 to represent the required endeavour to use DataTime for the development of a particular IPTS (see Section 5.3).

5.2.1 Spatial model and prediction configuration

Graph

The spatial model is defined as a classical directed graph structure as seen in the part II of Fig. 5.1, focusing on the `graph` package: a set of vertices that are nodes with individual characteristics, and a set of edges that are one way connections between two vertices (a bidirectional edge is simply represented with two directed edges). Graphs have a built-in `time_frame` attribute that represents the time-frame for which corresponding referential and historical data exist. Hence it is to the discretion of the designer to consider that changes amongst a graph structure yield a new graph, or if it just changes its `time_frame` by expanding or reducing it, or leaving it unchanged (e.g. when minor changes occur).

`Edge` and `vertex` characteristics (features) are made abstract in order to use any kind of data. The package `data` contains the different features and metrics definitions. In order to make those generic, we separated their identification from their actual data and type (following the type-object design pattern). This package contains three abstract classes that have to be specialized for a specific system: `Features`, `VolatileFeatures` and `Metrics`. `Features` represent characteristics of the edges and vertices. `VolatileFeatures` are characteristics that are not represented in the data, but that can be computed on the go (e.g bus line type extrapolated from its identifier, electrical cable resistance computed from its length, etc.). `Metrics` are data that are not characteristics, such as measures (speed, volume of water per hour, etc.) or time related information. Hence, data instances that extend those classes must be able to query data from the `Digital Shadow` (cf. 5.2.2).

The package `path` represents the abstraction of a path through the graph, that is, a sequence of at least one edge for the thinnest grain. `Paths` are abstract, hence one can easily create a path hierarchy if relevant for the targeted application domain (e.g

transportation networks, supply chains, smart-grids, drinking water networks, ...). Paths allow the building of hierarchical structures within the graphs while offering analysis and predictions scalability using micro-learning for the predictive aspects. Recently, micro-learning approaches have been successfully applied to graphs structures to provide what-if scenarios (e.g. in the context of power grid management [27]). In these approaches, specific models of local data are built instead of one large model on the overall data set. Micro-learning is typically useful for incremental predictive models, where one has to perform step-by-step decisions based on local properties, while yielding quality predictions. For instance for an IPTS, prediction of travel time or road reliability can be made using micro-learning over each edge and then aggregating the results using a specific composition law (sum for travel time, product for reliability, etc.).

In DataTime, data analysis and predictions are made at the level od edges (provided a predictive model has been configured for this purpose). For graphs on which data can be aggregated from edge level to different implementations of parent paths, the prediction or analysis at edge level will be automatically aggregated to higher levels with specific user defined composition laws (e.g speed of a bus between two bus stops, aggregated to the whole bus line, water leaks at single pipes sections, aggregated to the whole pipe, etc.).

Predictors

The `prediction` package contains the abstract class `Predictor` that embodies the predictive system of DataTime. It encapsulates predictive models and miscellaneous tools that define the expected behaviors of predictive models, from training to predicting. A specificity of this abstract class is that we included a way of managing the predictive models behavior through time. It proposes the following services:

— Choice of the predictive model, in order to choose the best suited predictive model depending on the prediction issue (using the strategy design pattern).

— Optimization of the model by searching for the best hyper-parameters . It embeds a grid search system that can either use a set of properties to seek from, or a random one (through the parameter `optimize` of `Predictor.train()` in Fig. 5.1).

— Training of the model, by feeding it with a training dataset, hyper-parameters and choosing whether to optimize the predictive model using grid search (through the parameters of `Predictor.train()` in Fig. 5.1).

— Saving and loading the predictors by transferring their data to the `Digital Shadow` interfaces (through the `Predictor.train()` method in Fig. 5.1).

— Model's health monitoring, that takes care of the predictive model ability to predict in a satisfying manner depending on its age and a prediction error threshold over which a new model is trained. This service can be disabled if needed (using the attribute `Predictor.activate_healthcare` in Fig. 5.1).
— Features extraction for prediction, that explores the edges features to make predictions (`Predictor.extractFeatures()` in Fig. 5.1).
— Predictions, by returning a `PredictedRun` when called (`Predictor.predict()` in Fig. 5.1).

To configure a predictor, a designer has to extend `Predictor` and implement the `prepareDoctorData()` method, whose work is to prepare data for the health check-up of the predictive model and the training of a new one when needed. The `doctor` is called when one wants to predict something and that the predictive model is more than `obsolescence` days old compared to the current system date. The objective behind that is to keep predictors up to date and efficient. We achieve this by comparing the residuals obtained predicting from a sample of the last available data from now to `back_days` back in time with their base error (the deviation of the predictive model obtained during the training phase). If the predictive models yield an average residual that is over `error_trigger` times the base error, then a new model is trained with the data yielded by `prepareDoctorData()`. The `doctor` can be deactivated if needed (for example when the models are not subject to derivation because, e.g. the data is quite consistent through time), by setting `activate_healthcare` to false.

`Predictors` can use different predictive models thanks to the abstract class `PredictiveModel`. If one uses a single machine learning API in which predictive models are children of a single interface, then any model of such an API should be made available by encapsulating them in an extension of `PredictiveModel`, by tweaking the `error()` method for each model (because the error computation varies, and the result is not always kept into a trained model), that is necessary for the `doctor()`. Of course, if one wants to code her own predictive model, the simplest way is to extend `PredictiveModel` and override the relevant methods.

Predictors are referred by edges only, due to the fact that our system relies on compositional prediction. In order to make any prediction with any machine learning model, predictors should have their own set of features, and prediction targets. Hence the edges that call the predictors are responsible for giving them the right features by using their inner method `getFeaturesVector` that transforms their features into a vector of doubles for the predictor (with an ad hoc encoding for categorical data). Note that if some exter-

nal features that have not been designed in the model have to be passed to the predictor, a container should be built to contain them. Predictors should then take the data they need directly within this object.

The training of predictors is made by the `Edge` class, that calls the method `train(...)` of `Predictor`. This method has to be fed with a dataframe containing the appropriate training data (i.e. in accordance with the predictor feature set, thereby the predictive model formula), obtained by querying the digital shadow, and a set of hyper-parameters through the parameters `prop`. The parameter `optimize` triggers a randomized grid search algorithm when set to true in order to find the best configuration for the predictive models hyper-parameters. Thanks to this, any machine learning model with any features set and prediction target can be trained and used at any path level in the framework.

Finally, the class `Graph` is responsible for the manipulation in a single graph instance such as seamless data analysis (through the method `getMetric(...)`), getting the corresponding historical or real-time data from the `Digital Shadow`), what-if scenarii (creating new edges, vertices, path and analysis against those new objects), and the orchestration of the data between the different parts of the model such as instantiating and feeding predictors with data when they need to be trained (through its edges). The main goal of this class is to provide a way to obtain a set of `runs` when one calls the method `getMetric(...)` by seamlessly returning historical or real time or predicted results from historical runs or predicted runs over the given `path` and time period passed to the method `getMetric(...)`. Note that it does not matter if there is one or more instances of graphs (e.g. a set of independent graphs distributed through time, with contiguous time-frames). In both cases it is the `getMetric(..)` method of `Digital Twin` is used by the end-user to explore data in time and space.

5.2.2 Digital twin/shadow description

The `Digital Shadow` is either the representation of an existing data environment on which DataTime can be plugged in a read-only manner in order not to have any side effect on the existing information system, or an actual fully managed data environment dedicated to DataTime, making it responsible for the management of all the data flows between the digital twin and the real system. The digital shadow should also be responsible for the saving of the graphs instances and predictors instances. Moreover, it has the responsibility of creating time-series used for data analysis while keeping them optimized and consistent through time. In short, any data that is yielded either by the

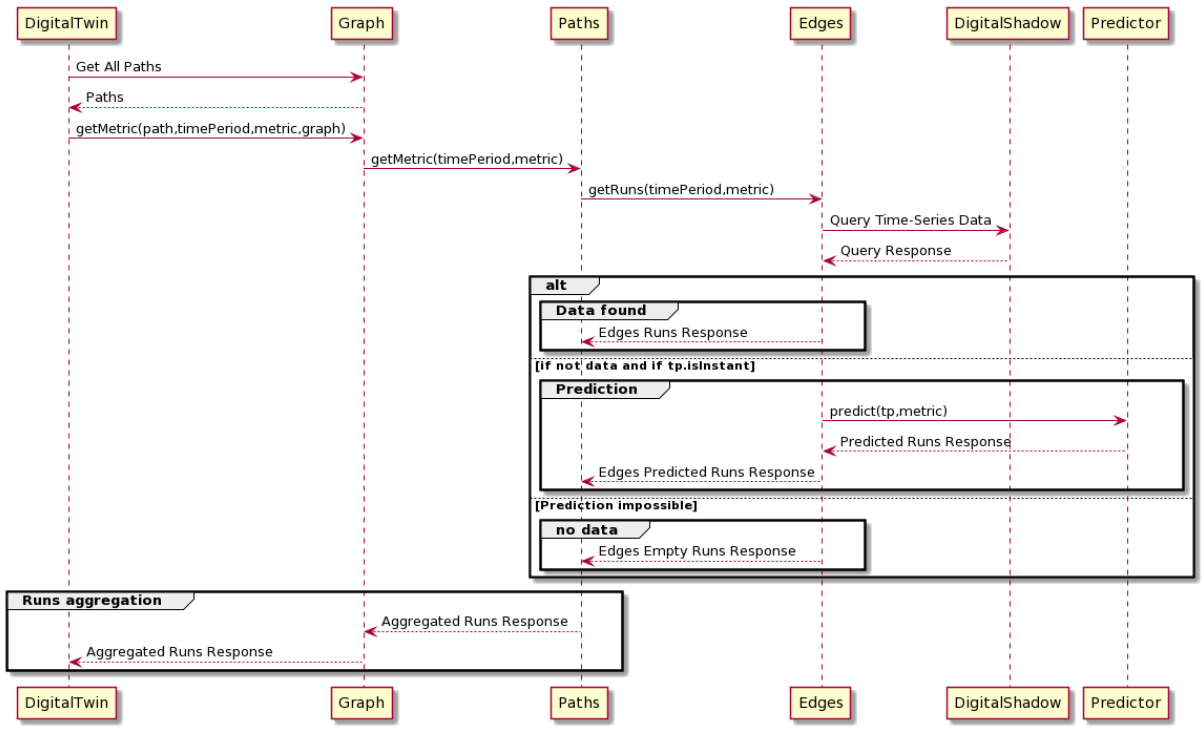

Figure 5.2 – Sequence diagram for getMetric

`Digital Twin` of DataTime or the real system sensors, etc. should transit and be governed by `Digital Shadow`. Fig. 5.2 shows an example of how data transits through the system when `getMetric(...)` is called.

The `Digital Twin` of DataTime represents the set of `graph` instances of the model, over the time. It contains the time representation through the class `TimePeriod` that embeds two timestamps. This class helps the representation of time frames and instants (when `periodStart == periodEnd`), and time manipulation in the framework. The time periods are used to request the set of graphs that are part of the class `Digital Twin`, when calling the method `getMetric(...)`, that returns a set of `runs` which represent the result of the queries made by the `Digital Twin` over its set of `graphs`. In other words, `runs` model the action of traveling through a `path` at a given time frame, or a specific event on a `path` for a given time frame, e.g. a bus traveling along a bus line, an amount of water traveling through a section of a pipe network, the loss of electrical current between two poles, etc. Runs are built using historical data. Hence they consist of different measures & metrics made on the network represented by an instance of the `graph`. Runs should be atomic, hence they should be unique and they should be bound to `edges` only, with runs

of longer paths built by aggregating runs of their `edges`. Runs can be historical / pseudo real-time (historical data) or predicted (using machine learning). Depending on the size of the historical data, runs should be lazily loaded when analyzed. However, saving the runs can be a bad idea if the historical data is huge: it would lead to storage starving in addition to duplicated data. However, this decision depends on the `Digital Shadow` structure. The `Digital Twin` class is also responsible for the building of graphs instances by querying the `Digital Shadow` for referential data, yielding `Graph` instances. Hence the `Digital Twin` is the end-user entry point on the framework.

5.3 Application to an Intelligent Public Transportation System

To assess the applicability of DataTime, we developed a Scala implementation of the framework adapted to urban bus networks.

IPTS such as modern urban bus networks are complex socio-technical systems made of hundreds of stakeholders, including humans, sensors, vehicles, information system, etc. Hence, a major concern with such an infrastructure is to gather, organize and normalize data, analysis and decisions in integrated tools.

IPTS are slowly evolving networks, yet if an important part of the bus lines are non changing for very long and continuous periods, there exists some variation within their structure, such as the changes on bus lines when long-term roadworks are planned, the creation of new bus lines, etc. Hence we considered that the `time_frame` of a single instance of the bus network is corresponding to the period that was defined by the operator (usually 1 to 2 weeks), during which there are no major modifications except for some day to day bus line deviations.

In our applicative environment, the information systems were already provided, also used by others. Hence, we plugged our framework on top of the existing `Digital Shadow` in order to avoid any side effect on it. In our implementation, the `Digital Twin` is responsible for the creation of as many graph instances corresponding to the different referential files that exist in the data. Referential files contain actual definitions of the bus network structure, as defined by the operator. i.e, if there are four sets of referential files, each of them defining a bus network that is valid for a given period of time, four different `graphs` instances will be created.

5.3.1 Graph and paths adaptation

We specialized DataTime for bus networks issues as shown in the bottom part of Fig. 5.1. The bus network is an extension of the class `Graph`, for which the longest path within it are `BusLines`, made of `Sections` that are an ordered collection of `InterStations` (Edges). The class `Vertex` is representing the bus stops (stations) in this context. All of the elements of the graphs are immutable, in order to keep data consistency when one wants to make, e.g. an analysis over a line/section/inter-station that exists in all the members of a set of graphs.

5.3.2 Features adaptation

`InterStations` contain `Features` such as length of the inter-station (in meters), type of road (one-way, two ways, reserved for buses, etc.), number of traffic lights, number of pedestrian crossings, etc. All those features are physical features, extracted from OpenStreetMaps [1] Section and line features are obtained by aggregating their respective subpart features. In order to manage features in the bus network implementation, we created `data instances` that extend the abstract classes from the package `data` (following the type-object design pattern). We then created several data features classes (e.g. `Length`, `TrafficSignalsCount`, etc.), one `VolatileFeature` class, called `LineType` and as many `Metrics` classes as needed amongst which `Speed`, `TravelTime`, etc.

5.3.3 Runs adaptation

If we take a look at `Run`, the data they correspond to in the `Digital Shadow` is the data gathered from bus trips all over the network. We describe trips as follows:

— Trip in edges: Bus trips from bus stop `origin` to bus stop `destination`;
— Trip in sections: Bus trips `origin` to bus stop `destination` within its ordered collection of edges;
— Trip in line: Bus trips from origin to destination (terminals).

Trips contain metrics such as start time, arrival time, travel time, speed, dwell time (for sub paths only). One could easily add any external feature such as smart card data, weather, traffic, etc.

1. https://gitlab.inria.fr/glyan/osm_bus_extractor

5.3.4 Data mass and lazy loading

Since an IPTS typically produces several GigaBytes of data per month, we soon end up with more data than can be managed on a local file system. We thus need a kind of lazy loading mechanism when querying trip data. Accordingly, runs are "built" on demand when a call is made through the method `getMetric()` from the `!DigitalTwin`. The resulting trips can be kept in memory up to a certain extent, but not saved. An advantage of lazy loading is that this allows the ingestion of data in pseudo-real time (e.g. when there is a need to observe what is going on in the bus network with a reasonable delay).

5.4 Experimentation at Keolis Rennes

5.4.1 Evaluation of the Predictive Model

The first stage in the deployment of DataTime at Keolis Rennes was to evaluate the precision of the predictive model, in order to build confidence into the whole framework. For that, we ran an experiment described as follows (outline of chapter 4):

1. We choose a set of 13 typical and different bus lines for which data could be gathered all along their path. This includes express, urban, inter-district and suburban bus lines;
2. We ran 13 experiments for each line, removing the respective line data from the training dataset that gathers data over year 2019;
3. We predicted the speed of each bus line using micro and macro-learning (i.e the training data consists of the set of full line trips data for macro-learning, while it is made of inter-station trips data for micro-learning);
4. We compared the aggregated micro-predictions and the macro-predictions to assess whether micro-learning is suitable for our model, and thus could be also used for unforeseen lines (e.g. in cases of traffic diversions).

Table 5.1 shows the averaged results of the experiments. It presents three prediction error measures for both macro and aggregated predictions. Those measures are the Root Mean Squared Error (RMSE) in km/h (eq. 5.1), Mean Absolute Error (MAE) in km/h (eq. 5.2) and Mean Absolute Percentage Error (MAPE) (eq. 5.3) in percent. All of these measures are the result of the comparison of the predictions with their actual counterparts,

hence, the lower the error, the better the prediction. Note that the RMSE is more sensitive to outliers than MAE and MAPE, which are not quadratic.

$$RMSE = \sqrt{\frac{1}{N}\sum_{1}^{N}(Y_t - \hat{Y}_t)^2} \qquad Y_t = actual, \hat{Y}_t = predicted \quad (5.1)$$

$$MAE = \frac{\sum_{1}^{N}(Y_t - \hat{Y}_t)}{N} \qquad Y_t = actual, \hat{Y}_t = predicted \quad (5.2)$$

$$MAPE = \frac{1}{N}\sum_{1}^{N}\left|\frac{Y_t - \hat{Y}_t}{Y_t}\right| * 100 \qquad Y_t = actual, \hat{Y}_t = predicted \quad (5.3)$$

For all measures, the precision obtained for the prediction is considered as good enough from the point of view of the bus operator (i.e Keolis Rennes). Further, the micro-learning approach is at least as good as macro learning approach [2] in terms of accuracy and offers state of the art performance [**berger-wolf_proceedings_2019**]. The three micro-learning approach predictions error indicators are lower than those of the macro learning approach. Hence this experiment validated the use of micro-learning predictions for bus networks such as the ones operated by Keolis in Rennes. Finally, the utmost interest of micro-learning is the possibility to predict at the finest possible grain within the bus network, hence build what-if scenarii such as the prediction of bus travel time on new or diverted bus lines.

Table 5.1 – Prediction accuracy

Aggregated-micro			**Macro**		
RMSE	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**
3.0	2.4	12.1%	3.2	2.9	12.2%

5.4.2 DataTime in Practice at Keolis

Our DataTime implementation was designed to be used by bus networks operators. Thanks to the framework we could develop the following features:

— The creation of new elements over the bus network, i.e. bus lines (or new bus lines sections), bus stops, inter-stations.

2. The full details of the validation experiments ia available at: https://gitlab.inria.fr/glyan/compred

— The analysis of any event on the network at any place and anytime in the bus network. For instance lines 1 to 20 in example 1 access past data using the method `getMetric()` with a time period located in the past
— The prediction of any metric on the network at any place and anytime in the bus network, e.g. Algorithm 1 lines 21 to 29 where the method `getMetric()` is call with a parameter meaning *now*
— Providing the operators hints on which detour should be applied on a bus line depending on, e.g. the expected speed of this detour. Even in a complex bus network with multiple possible paths between two bus stops, finding alternative possible routes is quite standard. It is typically solved with a search based greedy algorithm, parameterized with a maximum number of paths of a maximum length. The complex part is of course *evaluating* the suitability of the possible routes that the search algorithm would yield. Depending on the goal of the network operator (e.g. smallest travel time, best reliability, etc.), the system would then just have to pick the best path among the existing ones. As long as the system is able to predict a metric (speed, reliability) on any known or unknown edge, along any path, compositional prediction makes it possible to aggregate those predictions on any number of different paths allowing it to choose the best way to propose a detour, following one or more specific heuristics.
Algorithms 2 and 3 summarize the main steps of putting everything together to automatically apply a bus detour if the bus operator asks the system to.Algorithm 2 explains how to find all possible paths between two vertices in the graph, with some constraints on the number of iterations and paths found size. Algorithm 3 shows how to call algorithm 2 to yield an optimized deviation for a bus line, searching to maximize the speed of the deviation.

5.4.3 Lessons Learned

Performance and scalability

We fed our implementation of DataTime using Apache Spark v3.1.1. It was behaving as a datasink, making it possible to query a datalake containing more than 2 years of data, totaling nearly 40GB. As an example, the primo-execution of a query like the one visible at lines 1-9 in Algo.1, which consists of querying over all the trips of a bus line for a 2 years period, takes around 40 seconds on a computer equipped with a middle end 8 cores

Algorithm 1 Different call examples over the getMetric method

```
1: val busLine:Line = Graph.getLine("152", Direction.B)
2:
3: // TimePeriod for which historical data exists
4: var tp:TimePeriod = TimePeriod("2019-01-01 08:00:00","2021-02-01 00:00:00")
5: var metric:Metric = Speed
6:
7: // Will return all the runs found for line 152 over tp
8: var runs:Set[Run] = getMetric(busLine,tp,metric)
9: runs.foreach.displayMetric()
10:
11: // TimePeriod for real time Data
12: val now:Long = System.currentTimeMillis
13: tp.setPeriodStart(now)
14: tp.setPeriodEnd(now)
15:
16: /** Will return runs corresponding to the last data available
17: * in the digitalshadow, comparing with now
18: */
19: runs:Set[Run] = getMetric(busLine.getSections.head,tp,metric)
20: runs.foreach.displayMetric()
21:
22: // TimePeriod for future date
23: tp.setPeriodStart("2022-03-19 17:30:00")
24: tp.setPeriodEnd("2022-03-19 17:30:00")
25: metric = TravelTime
26:
27: // Will seamlessly return a predictedRun
28: runs:Set[Run] = getMetric(busLine.getInterStation(5),tp,metric)
29: runs.foreach.displayMetric()
```

x86 CPU. If one executes this query a second time, the result is almost instantaneous provided the last request results were kept in memory. In a nutshell, the bigger the period and the longer the path, the slower the querying will be. Hence, the performance and design of the `Digital Shadow` services are paramount for querying to be efficient.

The predictors, that are able of continuous learning (by automatically training new predictive models when needed), must be trained before being able to predict. The training time depends on many factors but there are 3 of them that will have a significant impact. Those are the size of the training dataset, the number of features and the hyper-parameters tuning. The latter can be the worst one if, e.g. one uses grid search to find the best set of hyper-parameters (using `optimize=true` in the method `train(...)`). We decided to train

models with a training dataset that contains 1 or 2 month of data, with a set of 8 features and the default set of hyper-parameters. The training of such a model takes no longer than 10 minutes on a middle end computer (8 cores, 16GB RAM). Hence, the continuous training if the predictive models eventually turn dull would be in the same time range. This is rather acceptable knowing that a single model is able to yield predictions for the whole system.

Finally, our implementation is scalable in those ways:

1. The digital shadow can rely on scalable databases, hence the storage and querying can be distributed amongst different machines if needed.

2. The implementation of the framework can be used on any machine and does not need a tremendous amount of computing power to analyze data or predict data: The data collection for analysis relies on the digital shadow ability to scale, and the predictive models training time is short even on a single machine.

Impact at Keolis

If we recall what we saw in section 1.1.2 of chapter 1, Keolis Rennes is in need of a tool to centralize data management, data analysis, and prospective/predictive models. Thereby, a framework such as DataTime favors the data centralizing with a focus on scalability for data analysis, making it possible for domain experts to integrate past, present and future within a traditional information system containing a priori models. Moreover, DataTime could be even more user friendly with the providing of future DSLs, for, e.g. facilitating the integration of new data sources, the edition of line topologies or make data analysis more fluid, etc.In addition, DataTime allowed us to highlight specificities of the bus networks that were yet not visible for the operator. Indeed, the use of predictive models allowed the analysis of feature importance, hence to do sensitivity tests for, e.g. speed and vehicle engine type. It appeared that the electrical buses that were test running on the bus line 12 for a few months were systematically slower than their combustion counterparts, which was later explained by their higher gravity center, due to the presence of the traction batteries on the buses roofs.

5.5 Conclusion and Perspectives

In this chapter we propose a framework called DataTime to implement models at run-time which capture the state of a socio-technical system according to the dimensions of both time and space. Space is modeled as a slowly evolving directed graph where both nodes and edges bear local and independent states (i.e, values of properties of interest). DataTime provides a unifying interface to query the past, present and future (predicted) states of such socio-technical system, hiding the complexity and scalability issues of dealing with huge time series and machine learning. We applied our framework in the context of an urban transportation system, and concretely deployed and evaluated it on the IPTS of the city of Rennes (France).

DataTime makes it possible for domain experts to integrate past, present and future within a traditional information system containing a priori models. Still up to now, using DataTime requires experts to work at the level of the chosen programming language, here Scala, which is seldom known to the IT department of an IPTS operator such as Keolis. We thus plan to make it easier to use the DataTime framework by providing a set of DSL embodying its main abstractions, thus facilitating the integration of new data sources, the edition of the network topology, or make data analysis more fluid.

As future work, we envision to apply our framework for other application domains that bear structural and temporal similarities with IPTS, such as smart grids, water abduction systems, or supply chains. As long as they fit the directed graph abstraction at the core of DataTime, and as long as global properties of interest could be obtained by composing atomic values associated to edges, we do not foresee any issue in specializing DataTime towards these systems.Thi

Algorithm 2 Detours finding (Scala inspired pseudo-code)

```
1: /** Call function **/
2: def detours(orig, dest, maxLength, maxDetours):
3: val detoursSet = new Set[Set[Edge]]()
4: /** Get the edges starting from orig **/
5: val possibleEdges:Set[Edge] = GRAPH.edges.filter(_.orig == orig)
6: val edgesSet = new Set[Edge]()
7: for (e: Edge in possibleEdges) do
8:    _detours(e, dest, 1, maxLength, maxDetours, detoursSet)
9: end for
10: return detoursSet
11: end detours
12:
13: /** Recursive function **/
14: private def _detours(currentEdge, dest, order, maxLength, maxDetours, edgesSet, detoursSet):

15: if (detoursSet.length < maxDetours and order < maxLength) then
16:    /** Non duplication check **/
17:    val vertexInSet:Boolean = if (edgesSet.isEmpty) false
18:    else edgesSet.map(_.orig == currentEdge.dest).reduce(_||_)
19:    if (currentEdge.dest == dest and not vertexInSet and edgesSet.add(currentEdge))
       then
20:       /** FINAL CASE **/
21:       detoursSet.add(edgesSet)
22:    else if (not edgesSet.contains(currentEdge) and not vertexInSet) then
23:       /** GREEDY CASE **/
24:       val possibleEdges:Set[Edge] = GRAPH.edges.filter(_.orig == currentEdge.dest)
25:       /** This loop should be parallel for better performance **/
26:       for (e:Edge in possibleEdges) do
27:          edgesSet.add(currentEdge)
28:          if (not edgesSet.contains(e)) then
29:             _detours(e, dest, order+1, maxLength, maxDetours, edgesSet.clone, detoursSet)
30:          end if
31:       end for
32:    end if
33: end if
34: end _detours
```

Algorithm 3 Example of how to get the best detour on line A2 between A and B for the morning rush hour of Tuesdays during working periods (Scala inspired pseudo-code)

```
1: val line:Line = GRAPH.lines.A2
2: val detourStart:Vertex = A
3: val detourEnd:Vertex = B
4: val originalRoute:Set[Edge] = line
5: .getSubRoute(detourStart,detourEnd)
6: val day = Days.TUESDAY
7: val holidays = Holidays.NONE
8: val period = Periods.MORNING_RUSH_HOUR
9: val metric = Metrics.SPEED
10: val possibleDetours:Set[Set[Edge]] =
11: detours(detourStart, detourEnd, 20, 10)
12: .except(originalRoute)
13: val metrics:Set[Double] = possibleDetours
14: .map(_.predict(metric,TimePeriod(holidays, day, period))
15: val bestDetour:(Set[Edge],Double) = possibleDetours
16: .zip(metrics).sortBy(_._2).last
17: GRAPH.enact(line,bestDetour)
```

Conclusion and future work

This chapter is a conclusion to this thesis. It summarizes the main contributions of this document and proposes some future work to tackle problems unsolved or raised by this work.

5.6 Conclusion

Urban Transportation Networks are going to get closer to data and computing technologies with time. Not only because the use of sensors becomes easier and cheaper thanks to better communication technologies and hardware production methods, but also because cities and operators are willing to optimize their transportation networks, keeping them efficient and attractive in upcoming smart-cities. However, all this will yield growing and tremendous amount of data, for the more the system is cyber-equipped, the more data is generated. Hence, more domain expert and IT engineers and researchers will be needed to manage, handle, analyze and produce interpretations of this huge heterogeneous data. Therefore, the building of data management systems, or even digital twins is desirable to help bus networks operators tackle the problem of integrating massive heterogeneous data, extract information from this data to yield quality analysis or predictions, and finally have a continuously available virtual representation of the bus network that makes it possible the exploration of data through space and time.

In the first contribution, we proposed to evaluate the data quality of the Automatic Vehicle Location (AVL) system of the bus network of the city of Rennes, France. Our findings are that this system is error prone, and that a specific data management and data qualification environment is needed for both production and research purposes. However, the data quality level must be bound to the specific need of the users of this data, in order not to over cleanse it, yielding potential high cleansing cost for minor quality improvements.

The second contribution makes use of massive heterogeneous data to assess the existence of factors that supposedly have an impact on bus commercial speed. We could gather data for traffic, bicycles, ridership, weather and roads infrastructure. We inves-

tigated in depth traffic and ridership thanks to the lock-down period that was imposed in France in spring 2020. We were able to assess the existence of impacting factors, but the confirmation of the genericity of these factors could not be reached. However we were able to raise a series of hypotheses that might be helpful for bus networks operators. Yet, those hypotheses have yet to be validated on further works.

The third contribution makes use of the two previous ones. We use the data we obtained using the processes described in the first contribution, and some of the factors identified in the second contribution in order to make so called micro-predictions at the inter-station level. We considered the bus network to be a directed graph, on which vertices are bus stops and edges inter-stations. We rely our system on the fact that, e.g. commercial speed over a given path is composed of the sum of edges lengths divided by the sum of edges travel time. We targeted the prediction of bus commercial speed on both known and unknown inter-stations in order to propose a system that seamlessly provides predictions and what-if scenarii analysis (e.g. creation of a new bus routes and bus lines). Finally, we have shown that micro-predictions are as good as predictions at terminal to terminal (whole bus line) scale, while offering compositional properties making it way more flexible for spatio-temporal predictions.

The fourth and final contribution proposes DataTime, a framework to implement models at runtime which capture the state of the system according to both time and space modeled as a directed graph. It is the direct outcome of the three previous contributions, using our findings as inputs. In DataTime, a transportation network is modeled as a slowly evolving directed graph where both nodes and edges bear local and independent states (i.e., values of properties of interest). DataTime provides a unifying interface to query the past, present and future (predicted) states of such a socio-technical system, hiding the complexity and scalability issues of dealing with huge time series and machine learning. We applied our framework in the context of an urban transportation system, and concretely deployed and evaluated it on the Public Transportation Information System (PTIS) of the city of Rennes (France).

In the end, our contributions aim at providing domain experts with methods for the integration of heterogeneous data, data analysis abilities and predictive models in monolithic frameworks. In addition, this work starts the depreciating of the use of multi-tools systems for data management.

5.7 Perspectives

5.7.1 Bus commercial speed impacting factors assertion and generalization

Our findings in Chapter 3 are a first step to formally identify factors that have an actual impact on the bus commercial speed by using real world data. However, the use of accurate and complete data is of need to strengthen the assumptions we raised. In particular, the importance of the different factors is probably city-dependant, hence factors that are prominent in Rennes can be minor in other cities. Hence a further work based on this one would be to lead this kind of survey in various cities over the world to discriminate factors that are global from those which are dependent of either a group of similar cities or a city in particular.

5.7.2 Prediction in graphs: compute minimum paths size over which micro-predictive models can be avoided

We saw in Chapter 4 that if micro-predictive models are as capable as predictive models that consider the longest paths of a given graph for inputs, there seems to be a path size threshold over which micro-predictive models should be avoided because of noise amplification yield by the aggregation rules. This can be considered as an operational research problem. For a given graph, a dedicated (greedy) algorithm should travel along all the available paths, and train a predictive model and compute residuals for every step in the paths. Doing this has at least two purposes. First it could be useful to assess whether micro-predictive models are indeed limited over a given path size, if we do not consider training data quality. Second, this could help to understand whether the so called threshold is to be shared amongst all the paths of the graph, or if there are many local thresholds, thereby enhancing the complexity of this issue.

5.7.3 DataTime domain specific language

The first shot of DataTime proposed in Chapter 5 of this thesis can be completed by integrating a Domain Specific Language (DSL). This DSL would be plugged on top of the core of the framework and propose a seamless way to, e.g.

— create/modify either the whole or its subparts;

— query the graph for data (i.e a query sub-DSL);
— add/remove/modify data sources;
— manipulate the predictive components of the framework.

The adjunction of an integrated domain specific language to DataTime would enhance the end-user experience, while preserving the genericity of the framework.

5.7.4 Data collection expansion for the current Intelligent Public Transportation System (IPTS) implementation of DataTime

Currently, our IPTS implementation of DataTime can get data from OpenStreetMaps for any bus line that has been reproduced on the online OSM servers by the community. A limitation however is that if one wants to create a new bus line in our DataTime IPTS implementation and get the road infrastructure data from online OSM, this new line has to be designed on OSM too. This is a problem because OSM is intended to be a virtual representation of the real world, thereby the design of non existing bus lines for experiment purpose on online OSM has to be avoided.

However, there might be a way to link DataTime to a local OSM server using osm2pgsql [3] (tool that populates postgresql with dumps from OSM) and OSMosis [4] (that synchronizes postgresql databases and the local server). The harmonization of those tools could make it possible to either modify bus lines directly from DataTime or even the local OSM server, the changes being automatically synchronized between one another. Furthermore, OSMosis could be used to continuously integrate new data from online OSM while keeping changes of the local OSM server safe. However, this would need a lot of work to minutely adapt the local OSM instance so that it keeps the modification one makes among its current database, while it should be able to pull new information from world wide OSM database without pushing anything to it nor failing because of data conflicts.

5.7.5 Meta-learning and extrapolation of aggregation rules from data for micro-predictive models

In the current state, as described in Chapter 5, our encapsulation of predictive models proposes a way to use any predictive model with integrated model health management.

3. https://osm2pgsql.org/
4. https://wiki.openstreetmap.org/wiki/Osmosis

The so called `doctor` is used to deprecate a predictive model and train a new one when it is not able to predict with a satisfying accuracy among the last data to date. This encapsulation could be further enhanced by adding meta-learning, as suggested in [28]. One other limitation of the micro-predictive model we proposed is that the pre-condition needed to use it is the existence of aggregation rules. Indeed, micro-predictive models are desirable to predict non-atomic models (that can be broken down into sub parts) only. However those aggregation rules used to re-conciliate predictions made at the lowest level to higher levels in the model have to be manually implemented in the tool that makes use of micro-predictive models. Finding a way to extrapolate the aggregation rules directly from the input data could help the integration of predictive models. Hence, the implementation of meta-learning systems along with aggregation rules extrapolation would make it possible for the predictive components of DataTime to become a fully autonomous system, thereby totally user friendly.

List of Figures

List of Tables

Appendix A

Suplementary material

.

remove l
final

A.1 Micro predictions charts

Figures A.1, A.2, A.3, A.4 and A.5 show broader results of the predictions results of the experiments led in chapter 4.

A.2 Miscellaneous

Figure A.1 – micro-predictions of bus line 532 with and without OSM features

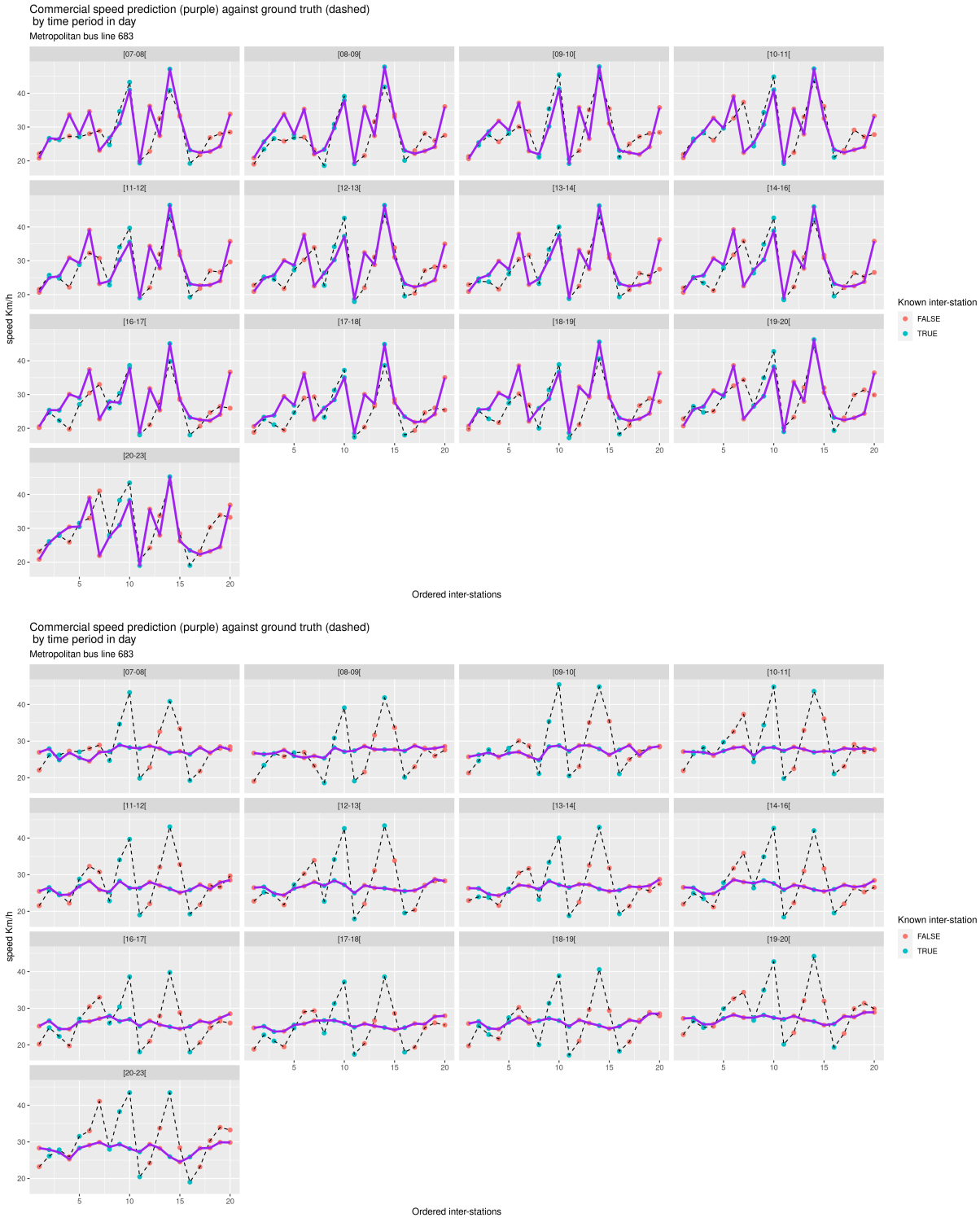

Figure A.2 – micro-predictions of bus line 683 with and without OSM features

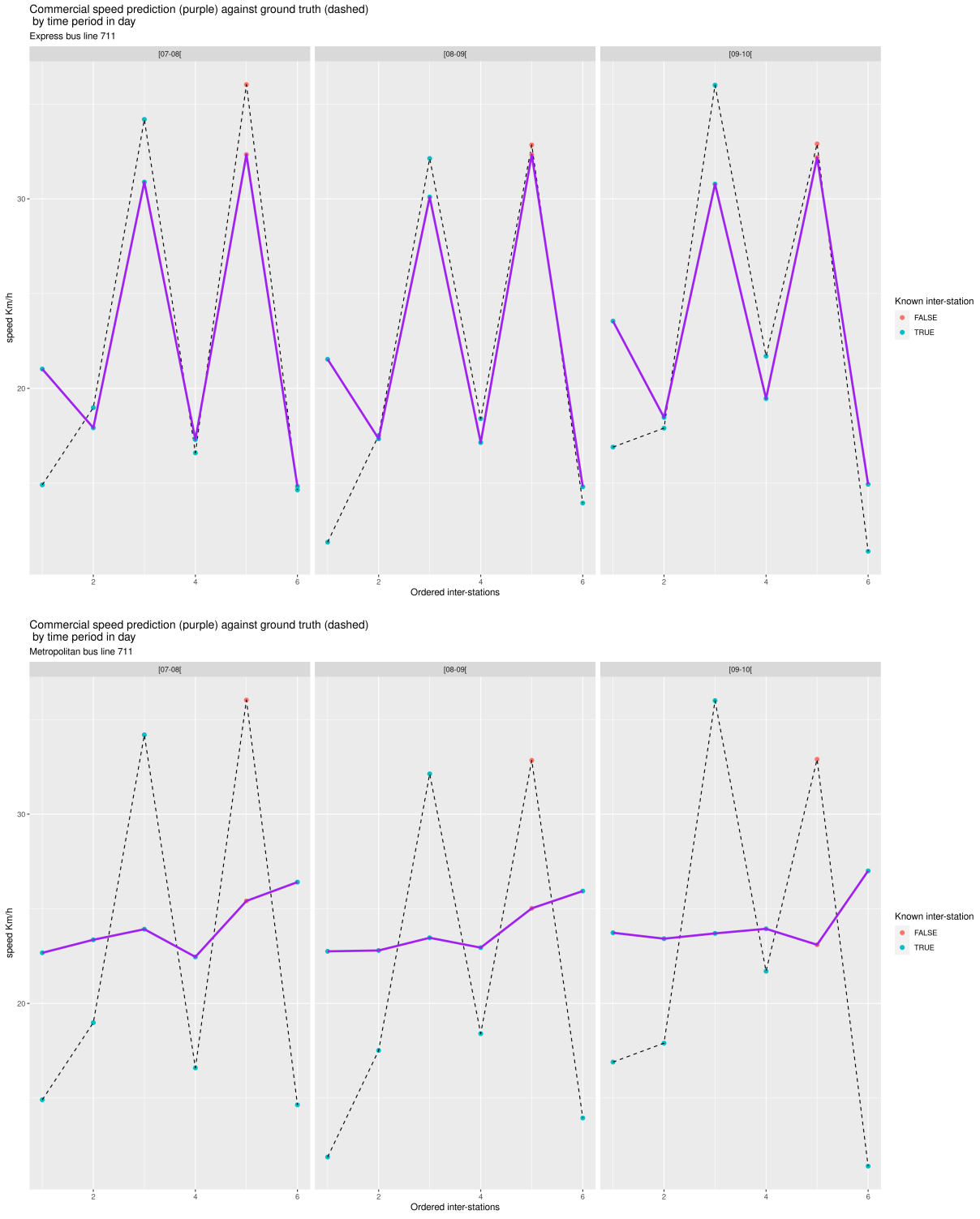

Figure A.3 – micro-predictions of bus line 711 with and without OSM features

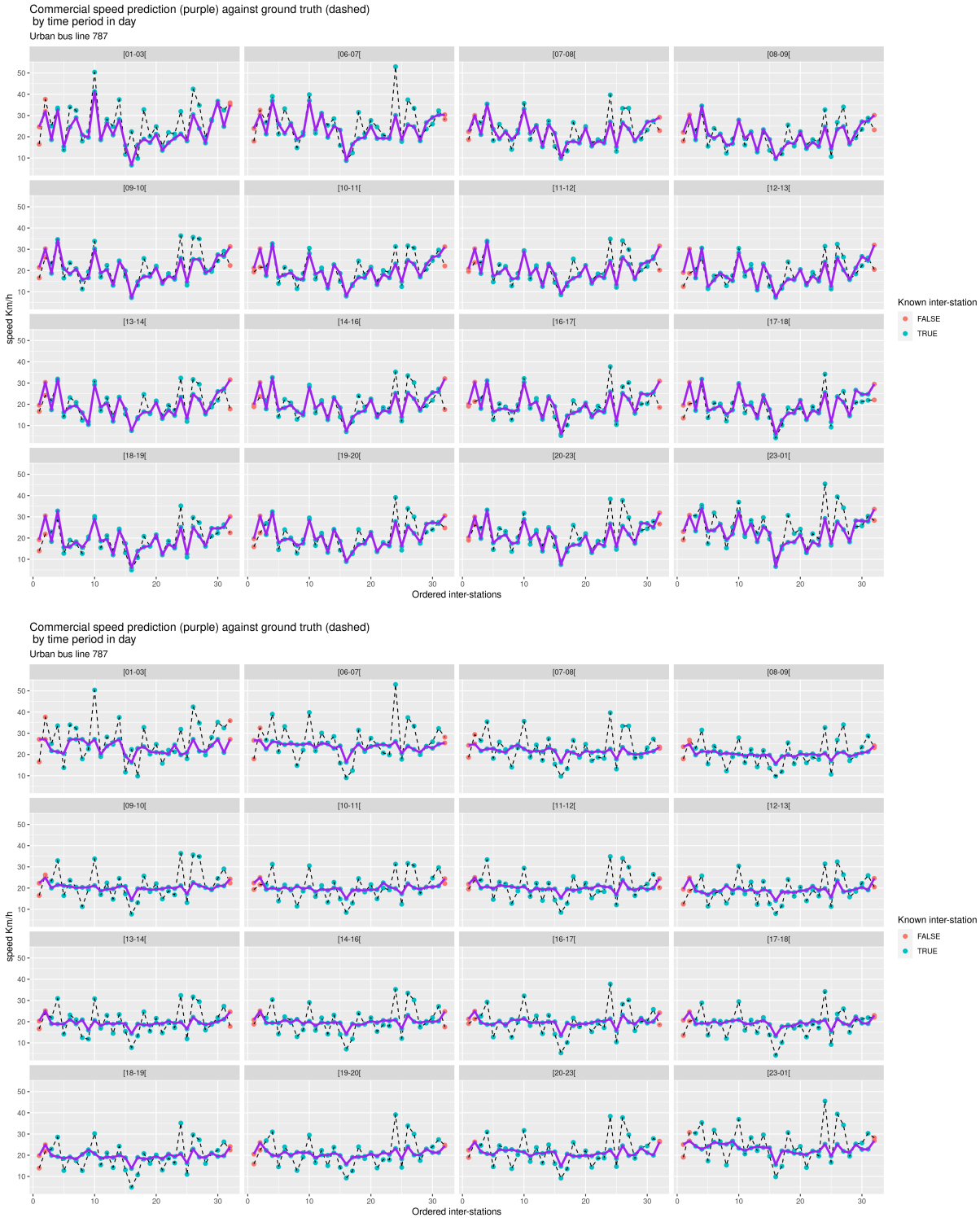

Figure A.4 – micro-predictions of bus line 787 with and without OSM features

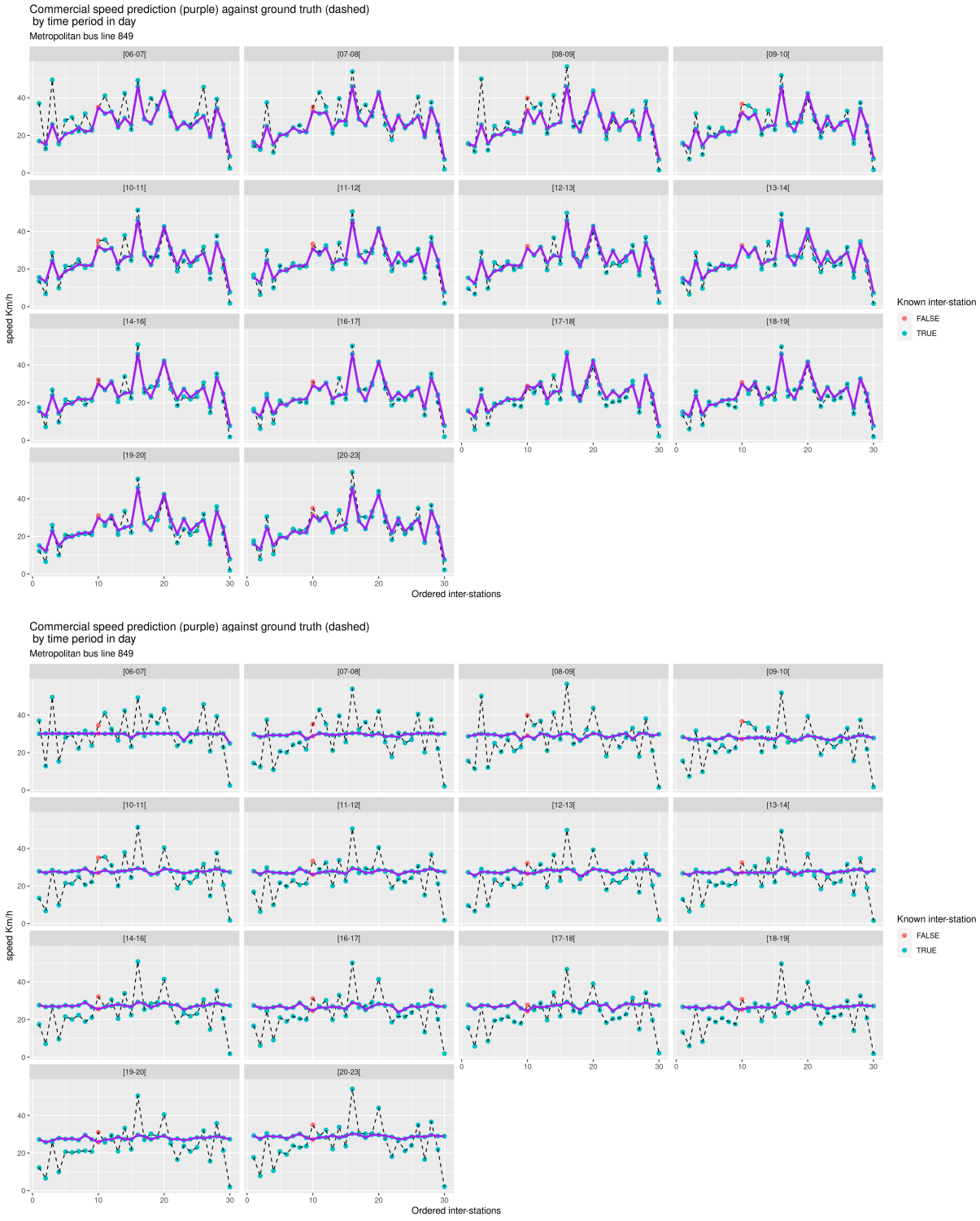

Figure A.5 – micro-predictions of bus line 849 with and without OSM features

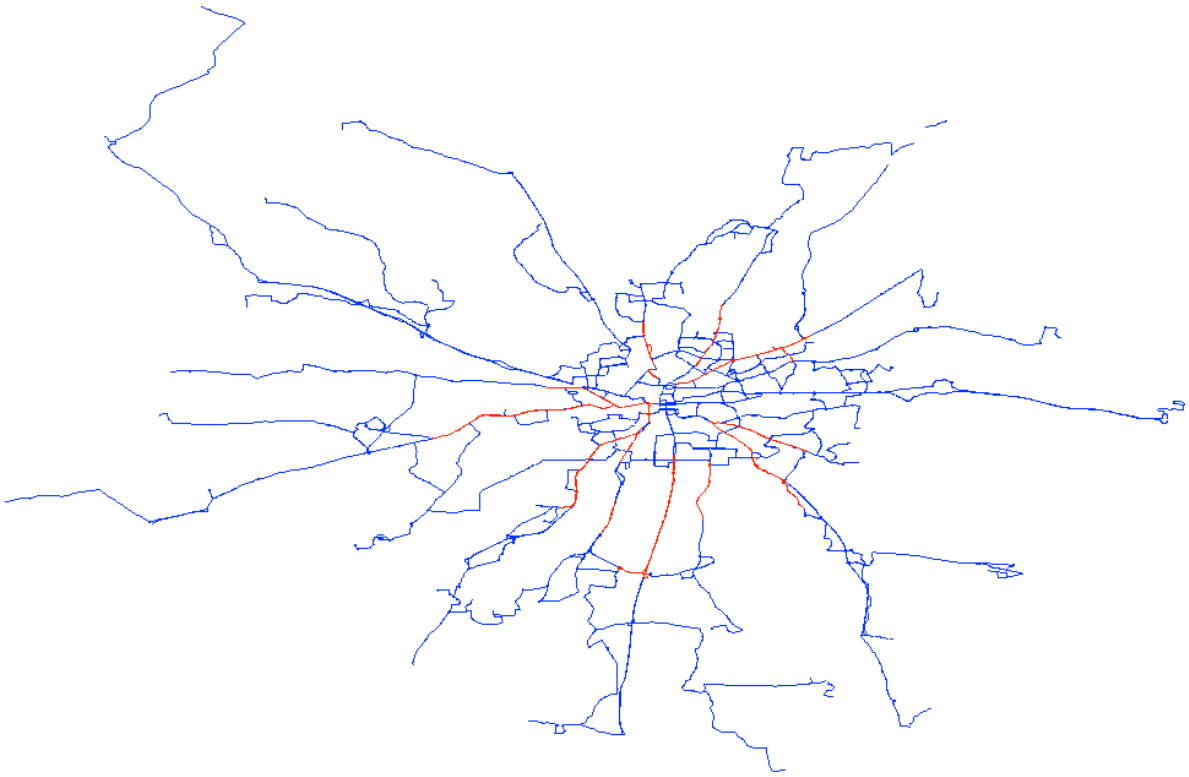

Figure A.6 – Geographical view of the STAR bus network drawn from the data extracted out of OpenStreetMap database dump. Red portions represent part of the network for which traffic data could be matched

BIBLIOGRAPHY

[1] Jeffrey L Adler and Victor J Blue, « Toward the design of intelligent traveler information systems », en, *in*: *Transportation Research Part C: Emerging Technologies* 6.*3* (June 1998), pp. 157–172, ISSN: 0968090X, DOI: 10.1016/S0968-090X(98)00012-6.

[2] O. Aloquili, A. Elbanna, and A. Al-Azizi, « Automatic vehicle location tracking system based on GIS environment », en, *in*: *IET Software* 3.*4* (2009), p. 255, ISSN: 17518806, DOI: 10.1049/iet-sen.2008.0048.

[3] Mehmet Altinkaya and Metin Zontul, « Urban Bus Arrival Time Prediction: A Review of Computational Models », en, *in*: 2.*4* (2013), p. 7.

[4] H. Amirat et al., « MyRoute: A Graph-Dependency Based Model for Real-Time Route Prediction », *in*: *JCM* 12 (2017), p. 668.

[5] Jaume Barceló et al., « A methodological approach combining macro, meso and micro simulation models for transportation analysis », en, *in*: (2005), p. 24.

[6] Christopher Barron, Pascal Neis, and Alexander Zipf, « A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis: A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis », en, *in*: *Transactions in GIS* 18.*6* (Dec. 2014), pp. 877–895, ISSN: 13611682, DOI: 10.1111/tgis.12073.

[7] Germà Bel and Maximilian Holst, « Evaluation of the impact of Bus Rapid Transit on air pollution in Mexico City », en, *in*: *Transport Policy* 63 (Apr. 2018), pp. 209–220, ISSN: 0967070X, DOI: 10.1016/j.tranpol.2018.01.001.

[8] Jean-Yves Blais, Jacques Lamont, and Marc Rousseau, « The HASTUS Vehicle and Manpower Scheduling System at the Société de transport de la Communauté urbaine de Montréal », en, *in*: *Interfaces* 20.*1* (Feb. 1990), pp. 26–42, ISSN: 0092-2102, 1526-551X, DOI: 10.1287/inte.20.1.26.

[9] Francis Bordeleau et al., « Towards Model-Driven Digital Twin Engineering: Current Opportunities and Future Challenges », *in*: *ICSMM 2020 - International Conference on Systems Modelling and Management*, Bergen, Norway, June 2020.

[10] Oded Cats et al., « Mesoscopic Modeling of Bus Public Transportation », en, *in*: *Transportation Research Record: Journal of the Transportation Research Board* 2188.*1* (Jan. 2010), pp. 9–18, ISSN: 0361-1981, 2169-4052, DOI: 10.3141/2188-02.

[11] Avishai Ceder and Nigel H.M. Wilson, « Bus network design », en, *in*: *Transportation Research Part B: Methodological* 20.*4* (Aug. 1986), pp. 331–344, ISSN: 01912615, DOI: 10.1016/0191-2615(86)90047-0.

[12] Vítor Cerqueira et al., « Arbitrated Ensemble for Time Series Forecasting », *in*: *Machine Learning and Knowledge Discovery in Databases*, ed. by Michelangelo Ceci et al., Cham: Springer, 2017, pp. 478–494, ISBN: 978-3-319-71246-8.

[13] Benoit Combemale et al., « A Hitchhiker's Guide to Model-Driven Engineering for Data-Centric Systems », en, *in*: *IEEE Software* (2020), ISSN: 0740-7459, 1937-4194, DOI: 10.1109/MS.2020.2995125.

[14] Cristian E. Cortés et al., « Commercial bus speed diagnosis based on GPS-monitored data », en, *in*: *Transportation Research Part C: Emerging Technologies* 19.*4* (Aug. 2011), pp. 695–707, ISSN: 0968090X, DOI: 10.1016/j.trc.2010.12.008.

[15] Xavier Courtois and Frédéric Dobruszkes, « L'(in)efficacité des trams et bus à Bruxelles, une analyse désagrégée », fr, *in*: *Brussels Studies. La revue scientifique électronique pour les recherches sur Bruxelles / Het elektronisch wetenschappelijk tijdschrift voor onderzoek over Brussel / The e-journal for academic research on Brussels* (June 2008), ISSN: 2031-0293, DOI: 10.4000/brussels.603.

[16] Teresa Cristóbal et al., « Bus Travel Time Prediction Model Based on Profile Similarity », en, *in*: *Sensors* 19.*13* (June 2019), p. 2869, ISSN: 1424-8220, DOI: 10.3390/s19132869.

[17] J. Deng, H. Nie, and C. Chen, « Research on Bus Passenger Traffic Forecasting Model based on GPS and IC Card Data », *in*: *in Proceedings of the 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)* (2019).

[18] Sabeur Elkosantini and Saber Darmoul, « Intelligent Public Transportation Systems: A review of architectures and enabling technologies », en, *in*: *2013 International Conference on Advanced Logistics and Transport*, Sousse, Tunisia: IEEE, May 2013, pp. 233–238, DOI: 10.1109/ICAdLT.2013.6568465.

[19] J. Esser and M. Schreckenberg, « Microscopic Simulation of Urban Traffic Based on Cellular Automata », en, *in*: *International Journal of Modern Physics C* 08.*05* (Oct. 1997), pp. 1025–1036, ISSN: 0129-1831, 1793-6586, DOI: 10.1142/S0129183197000904.

[20] Reza Zanjirani Farahani, « A review of urban transportation network design problems », en, *in*: *European Journal of Operational Research* (2013), p. 22.

[21] Dan Feldman, Melanie Schmidt, and Christian Sohler, « Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering », en, *in*: *SIAM Journal on Computing* 49.*3* (Jan. 2020), pp. 601–657, ISSN: 0097-5397, 1095-7111, DOI: 10.1137/18M1209854.

[22] Martin Fellendorf, « VISSIM:A microscopic Simulation Tool to Evaluate Actuated Signal Control including Bus Priority », *in*: *64th Institute of Transportation Engineers Annual meeting* (Oct. 1994).

[23] R. Fernandez and E. Valenzuela, « A model to predict bus commercial speed », *in*: *Traffic Engineering & Control* 44.*2* (Feb. 2003), ISSN: 0041-0683.

[24] Jean-François Girres and Guillaume Touya, « Quality Assessment of the French OpenStreetMap Dataset: Quality Assessment of the French OpenStreetMap Dataset », *in*: *Transactions in GIS* 14.*4* (Aug. 2010), pp. 435–459, ISSN: 13611682, DOI: 10.1111/j.1467-9671.2010.01203.x.

[25] Tim Godfrey et al., « Modeling Smart Grid Applications with Co-Simulation », en, *in*: *2010 First IEEE International Conference on Smart Grid Communications*, Gaithersburg, MD, USA: IEEE, Oct. 2010, pp. 291–296, ISBN: 978-1-4244-6510-1, DOI: 10.1109/SMARTGRID.2010.5622057.

[26] Mehmet Akif Gormez, Ehsanul Haque, and Yilmaz Sozer, « Cost Optimization of an Opportunity Charging Bus Network », en, *in*: *IEEE Transactions on Industry Applications* (2021), pp. 1–1, ISSN: 0093-9994, 1939-9367, DOI: 10.1109/TIA.2021.3061031.

[27] Thomas Hartmann et al., « GreyCat: Efficient what-if analytics for data in motion at scale », en, *in*: *Information Systems* 83 (July 2019), pp. 101–117, ISSN: 03064379, DOI: 10.1016/j.is.2019.03.004.

[28] Thomas Hartmann et al., « Meta-Modelling Meta-Learning », en, *in*: *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Munich, Germany: IEEE, Sept. 2019, pp. 300–305, ISBN: 978-1-72812-536-7, DOI: 10.1109/MODELS.2019.00014.

[29] Thomas Hartmann et al., « The Next Evolution of MDE: A Seamless Integration of Machine Learning into Domain Modeling », en, *in*: *Software & Systems Modeling volume* (2019), p. 17, DOI: 10.1007/s10270-017-0600-2.

[30] M. Hilbert and P. Lopez, « The World's Technological Capacity to Store, Communicate, and Compute Information », en, *in*: *Science* 332.*6025* (Apr. 2011), pp. 60–65, ISSN: 0036-8075, 1095-9203, DOI: 10.1126/science.1200970.

[31] Odd André Hjelkrem et al., « A battery electric bus energy consumption model for strategic purposes: Validation of a proposed model structure with data from bus fleets in China and Norway », en, *in*: *Transportation Research Part D: Transport and Environment* 94 (May 2021), p. 102804, ISSN: 13619209, DOI: 10.1016/j.trd.2021.102804.

[32] Markus Hofmann and Margaret O'Mahony, « The Impact of Adverse Weather Conditions on Urban Bus Performance Measures – an Analysis Using ITS Technology », en, *in*: *Intelligent Transportation Systems*, IEEE, 2005, p. 7.

[33] Z.D. Huang et al., « A GIS-based framework for bus network optimization using genetic algorithm », en, *in*: *Annals of GIS* 16.*3* (Nov. 2010), pp. 185–194, ISSN: 1947-5683, 1947-5691, DOI: 10.1080/19475683.2010.513152.

[34] O.J. Ibarra-Rojas et al., « Planning, operation, and control of bus transport systems: A literature review », en, *in*: *Transportation Research Part B: Methodological* 77 (July 2015), pp. 38–75, ISSN: 01912615, DOI: 10.1016/j.trb.2015.03.002.

[35] Sakari Jäppinen, Tuuli Toivonen, and Maria Salonen, « Modelling the potential effect of shared bicycles on public transport travel times in Greater Helsinki: An open data approach », en, *in*: *Applied Geography* 43 (Sept. 2013), pp. 13–24, ISSN: 01436228, DOI: 10.1016/j.apgeog.2013.05.010.

[36] Jerald Jariyasunant, « Improving Traveler Information and Collecting Behavior Data with Smartphones », En, PhD thesis, Berkeley University, 2012.

[37] Sami Kaivonen and Edith C.-H. Ngai, « Real-time air pollution monitoring with sensors on city bus », en, *in*: *Digital Communications and Networks* 6.*1* (Feb. 2020), pp. 23–30, ISSN: 23528648, DOI: 10.1016/j.dcan.2019.03.003.

[38] Jörg Christian Kirchhof et al., « Model-driven Digital Twin Construction: Synthesizing the Integration of Cyber-Physical Systems with Their Information Systems », en, *in*: (2020), p. 12.

[39] Yashaswi Kotagiri and Srinivas S. Pulugurtha, « Modeling Bus Travel Delay and Travel Time for Improved Arrival Prediction », *in*: *International Conference on Transportation and Development 2016*, American Society of Civil Engineers, June 2016, DOI: 10.1061/9780784479926.052.

[40] Gauthier Lyan et al., *Impact of Data Cleansing for Urban Bus Commercial Speed Prediction*, Research Report, Université de Rennes ; IRISA ; Keolis Rennes, May 2021.

[41] Xiaolei Ma and Xi Chen, « Public Transportation Big Data Mining and Analysis », en, *in*: *Data-Driven Solutions to Transportation Problems*, Elsevier, 2019, pp. 175–200, ISBN: 978-0-12-817026-7, DOI: 10.1016/B978-0-12-817026-7.00007-2.

[42] RAPHAEL N MARKELLOS, « Evaluating public transport efficiency with neural network models », en, *in*: *Transportation Research Part C: Emerging Technologies* 5 (1996), p. 12.

[43] Takuya Matsumoto, Kazutoshi Sakakibara, and Hisashi Tamaki, « Bus line optimization using multi-agent simulation model of urban traffic behavior of inhabitants applying branch and bound techniques », en, *in*: IEEE, July 2015, pp. 234–239, ISBN: 978-4-907764-48-7, DOI: 10.1109/SICE.2015.7285551.

[44] Ehsan Mazloumi et al., « An Integrated Framework to Predict Bus Travel Time and Its Variability Using Traffic Flow Data », en, *in*: *Journal of Intelligent Transportation Systems* 15.*2* (May 2011), pp. 75–90, ISSN: 1547-2450, 1547-2442, DOI: 10.1080/15472450.2011.570109.

[45] João Mendes-Moreira and Mitra Baratchi, « Reconciling Predictions in the Regression Setting: An Application to Bus Travel Time Prediction », en, *in*: *Advances in Intelligent Data Analysis XVIII*, ed. by Michael R. Berthold, Ad Feelders, and Georg Krempl, vol. 12080, Series Title: Lecture Notes in Computer Science, Cham:

Springer International Publishing, 2020, pp. 313–325, DOI: 10.1007/978-3-030-44584-3_25.

[46] Patrick Mikalef et al., « The human side of big data: Understanding the skills of the data scientist in education and industry », en, *in*: *2018 IEEE Global Engineering Education Conference (EDUCON)*, Tenerife: IEEE, Apr. 2018, pp. 503–512, ISBN: 978-1-5386-2957-4, DOI: 10.1109/EDUCON.2018.8363273.

[47] Brice Morin et al., « Models at Runtime to Support Dynamic Adaptation », *in*: *Computer* (2009), pp. 46–53.

[48] Madzlan Napiah and Ibrahim Kamaruddin, « ARIMA models for bus travel time prediction », en, *in*: 71 (2010), p. 10.

[49] Mauricio A Herna Ndez and Salvatore J Stolfo, « Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem », en, *in*: *Data mining and knowledge discovery* (1998), p. 29.

[50] Fátima Trindade Neves, Miguel de Castro Neto, and Manuela Aparicio, « The impacts of open data initiatives on smart cities: A framework for evaluation and monitoring », en, *in*: *Cities* 106 (Nov. 2020), p. 102860, ISSN: 02642751, DOI: 10.1016/j.cities.2020.102860.

[51] Maarten Ottens et al., « Modelling infrastructures as socio-technical systems », en, *in*: *International Journal of Critical Infrastructures* 2.*2/3* (2006), p. 133, ISSN: 1475-3219, 1741-8038, DOI: 10.1504/IJCIS.2006.009433.

[52] S. B. Pattnaik, S. Mohan, and V. M. Tom, « Urban Bus Transit Route Network Design Using Genetic Algorithm », en, *in*: *Journal of Transportation Engineering* 124.*4* (July 1998), pp. 368–375, ISSN: 0733-947X, 1943-5436, DOI: 10.1061/(ASCE)0733-947X(1998)124:4(368).

[53] Marie-Pier Pelletier, Martin Trépanier, and Catherine Morency, « Smart card data use in public transit: A literature review », en, *in*: *Transportation Research Part C: Emerging Technologies* 19.*4* (Aug. 2011), pp. 557–568, ISSN: 0968090X, DOI: 10.1016/j.trc.2010.12.003.

[54] Robert Gilmore Pontius, Olufunmilayo Thontteh, and Hao Chen, « Components of information for multiple resolution comparison between maps that share a real variable », en, *in*: *Environmental and Ecological Statistics* 15.*2* (June 2008), pp. 111–142, ISSN: 1352-8505, 1573-3009, DOI: 10.1007/s10651-007-0043-y.

[55] PPIAF and Wolrd Bank Group, *Factors Influencing Bus System Efficiency*, URL: https://ppiaf.org/sites/ppiaf.org/files/documents/toolkits/UrbanBusToolkit/assets/1/1d/1d.html.

[56] Stephen Riter and Jan McCoy, « Automatic Vehicle Location - An Overview », *in*: *IEEE Transactions on vehicular technology* (1977).

[57] Steve Robinson et al., « Methods for pre-processing smartcard data to improve data quality », en, *in*: *Transportation Research Part C: Emerging Technologies* 49 (Dec. 2014), pp. 43–58, ISSN: 0968090X, DOI: 10.1016/j.trc.2014.10.006.

[58] Günter Ropohl and Society for Philosophy and Technology, « Philosophy of Socio-Technical Systems », en, *in*: *Society for Philosophy and Technology Quarterly Electronic Journal* 4.*3* (1999), pp. 186–194, ISSN: 1091-8264, DOI: 10.5840/techne19994311.

[59] Tobias Schoenherr and Cheri Speier-Pero, « Data Science, Predictive Analytics, and Big Data in Supply Chain Management: Current State and Future Potential », en, *in*: *Journal of Business Logistics* 36.*1* (Mar. 2015), pp. 120–132, ISSN: 07353766, DOI: 10.1111/jbl.12082.

[60] Brian Smith, William Scherer, and James Conklin, « Exploring Imputation Techniques for Missing Data in Transportation Management Systems », en, *in*: *Transportation Research Record: Journal of the Transportation Research Board* 1836 (Jan. 2003), pp. 132–142, ISSN: 0361-1981, DOI: 10.3141/1836-17.

[61] Chuanwang Sun et al., « Urban public transport and air quality: Empirical study of China cities », en, *in*: *Energy Policy* 135 (Dec. 2019), p. 110998, ISSN: 03014215, DOI: 10.1016/j.enpol.2019.110998.

[62] Ben Taskar et al., « Link prediction in relational data », *in*: *Advances in neural information processing systems* 16 (2003), pp. 659–666.

[63] Telang, Samir et al., « Intelligent Transport System for a Smart City », en, *in*: *Security and Privacy Applications for Smart City Development*, 2021, pp. 171–187, DOI: 10.1007/978-3-030-53149-2_9.

[64] Paul R. Tétreault and Ahmed M. El-Geneidy, « Estimating bus run times for new limited-stop service using archived AVL and APC data », en, *in*: *Transportation Research Part A: Policy and Practice* 44.*6* (July 2010), pp. 390–402, ISSN: 09658564, DOI: 10.1016/j.tra.2010.03.009.

[65] Alejandro Tirachini, « Bus dwell time: the effect of different fare collection systems, bus floor level and age of passengers », en, *in*: *Transportmetrica A: Transport Science* 9.*1* (Jan. 2013), pp. 28–49, ISSN: 2324-9935, 2324-9943, DOI: 10.1080/18128602.2010.520277.

[66] Alejandro Tirachini, « Estimation of travel time and the benefits of upgrading the fare payment technology in urban bus services », en, *in*: *Transportation Research Part C: Emerging Technologies* 30 (May 2013), pp. 239–256, ISSN: 0968090X, DOI: 10.1016/j.trc.2011.11.007.

[67] Alejandro Tirachini and David A. Hensher, « Bus congestion, optimal infrastructure investment and the choice of a fare collection system in dedicated bus corridors », en, *in*: *Transportation Research Part B: Methodological* 45.*5* (June 2011), pp. 828–844, ISSN: 01912615, DOI: 10.1016/j.trb.2011.02.006.

[68] Wichai Treethidtaphat, Pattara-Atikom Wasan, and Sippakorn Khaimook, « Bus Arrival Time Prediction at Any Distance of Bus Route Using Deep Neural Network Model », *in*: *International Conference On Intelligent Transportation* (2017).

[69] Alejandra Valencia and Rodrigo Fernandez, « A method to calculate commercial speed on bus corridors », en, *in*: *Traffic Engineering & Control* (June 2012), p. 8.

[70] Lidong Wang, « Heterogeneous Data and Big Data Analytics », en, *in*: *Automatic Control and Information Sciences* (2017), p. 8.

[71] Di Weng et al., « Towards Better Bus Networks: A Visual Analytics Approach », en, *in*: *arXiv:2008.10915 [cs]* (Sept. 2020), arXiv: 2008.10915.

[72] Sun Yidan et al., *Bus Travel Speed Prediction using Attention Network of Heterogeneous Correlation Features*, en, ed. by Tanya Berger-Wolf and Nitesh Chawla, Philadelphia, PA: Society for Industrial and Applied Mathematics, May 2019, ISBN: 978-1-61197-567-3, DOI: 10.1137/1.9781611975673.

[73] Bin Yu et al., « Optimizing bus transit network with parallel ant colony algorithm », en, *in*: *Proceedings of the Eastern Asia Society for Transportation Studies* 5 (2005), p. 16.

[74] Liu Yu et al., « Calibration of Vissim for Bus Rapid Transit Systems in Beijing Using GPS Data », en, *in*: *Journal of Public Transportation* 9.*3* (July 2006), pp. 239–257, ISSN: 1077-291X, 2375-0901, DOI: 10.5038/2375-0901.9.3.13.

[75] M Zaki et al., « Online Bus Arrival Time Prediction Using Hybrid Neural Network and Kalman filter Techniques », en, *in*: 3.*4* (2013), p. 7.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Mobilité urbaine : Apprentissage automatique pour la construction de simulateurs à l'aide de masses de données.

Mot clés : Systèmes Intelligents pour les Transports Publics, Science des données, Apprentissage automatique, Jumeaux digitaux

Résumé : L'ère des données dans laquelle nous sommes entrés s'accompagne d'une explosion de ces dernières, tant en variété qu'en quantité. Le transport public est un domaine qui génère des données en masse, et les systèmes d'information sont souvent soutenus par des technologies anciennes qui peinent à maintenir l'existant en place alors que la quantité de données augmente continuellement. Ceci pose deux problèmes. Premièrement, les données massives générées par le réseau de transport doivent être qualifiées et enrichies avec des sources de données externes afin d'être utilisées pour la prise de décision. Deuxièmement, afin de limiter le nombre d'outils et la complexité de maintenance, il est souhaitable d'intégrer la gouvernance des données avec les outils d'aide à la décision pour permettre aux opérateurs non experts de manipuler ces données. A travers quatre contributions aboutissant à la proposition d'un cadre technique qui intègre le passé, le présent et le futur dans un système d'information traditionnel contenant des modèles a priori, cette thèse défend que l'intégration de divers ensembles de données hautement qualifiés provenant du monde réel dans un modèle spatio-temporel unique offre un moyen qualitatif, efficace et peu coûteux de faire des analyses, des prédictions et d'aider à la prise de décisions stratégiques pour les réseau de bus tout en dépréciant par ailleurs l'utilisation de systèmes de gestion des données au format multi-outils non intégrés.

Title: Urban mobility: Leveraging machine learning and data masses for the building of simulators

Keywords: Intelligent Public Transportation Systems, Data Science, Machine Learning, Digital Twins

Abstract: The so called data era we have entered in is accompanied by an explosion of data, both in variety and quantity. Public transportation is a data-intensive field, and related information systems are often supported by old technologies that struggle to keep up as the amount of data continually increases. This poses two problems. First, the massive data generated by the transportation network must be qualified and enriched with external data sources in order to be used for decision making. Second, in order to limit the number of tools and the complexity of maintenance, it is desirable to integrate data governance with decision support tools to allow non-expert operators to manipulate this data. Through four contributions leading to the proposal of a technical framework that integrates the past, present and future into a traditional information system containing a priori models, this thesis argues that the integration of various highly qualified datasets from the real world into a single spatio-temporal model provides a qualitative, efficient and low-cost mean of analysis, prediction and strategic decision support for bus networks while depreciating the use of data management systems in a non integrated multi-tool data management systems?